INSTITUTO POLITÉCNICO DE BRAGANÇA
Escola Superior de Tecnologia e Gestão

Plataforma de Gestão e Previsão de Produção Alimentar

Alexandre Ximenes - a33059

Dissertação apresentada à Escola Superior de Tecnologia e Gestão no âmbito do Mestrado em Informatica.

Trabalho orientado por:

Prof. Rui Pedro Lopes

Esta dissertação não inclui as críticas e sugestões feitas pelo Júri.

Bragança

2023-2024

Plataforma de Gestão e Previsão de Produção Alimentar

Alexandre Ximenes - a33059

Dissertação apresentada à Escola Superior de Tecnologia e Gestão no âmbito do Mestrado em Informatica.

Trabalho orientado por:

Prof. Rui Pedro Lopes

Esta dissertação não inclui as críticas e sugestões feitas pelo Júri.

Bragança

2023-2024

Dedicação

Dedico este relatório de projeto aos meus pais, que com grande esforço e dedicação tornaram possível que eu chegasse a este ponto, e à minha família, pelo apoio incondicional ao longo desta jornada....

Reconhecimento

Este Projeto foi apoiado pelo Professor Rui Pedro Lopes, cuja orientação e assistência foram fundamentais para o sucesso deste trabalho. Além disso, expresso minha gratidão à Fundação Klibur Matadalan (FKMD), que, com seu apoio, possibilitou que eu chegasse até aqui. Agradeço também aos meus colegas, cujo apoio e colaboração ao longo do curso foram inestimáveis.…

Abstract

The production of food is a business model that utilizes resources such as space, feed, or other consumables to provide sustenance, typically for humans. This work aims to study and develop a food production management and forecasting system, focusing primarily on dairy production, allowing users to manage production and access useful indicators for company operations, such as Input Income Over Feed Cost (IOFC), and forecasting costs/-revenues based on financial market trends. The objective is to develop a mobile platform with a backend accessible via Representational State Transfer (REST)/Javascript Object Notation (JSON) for managing and forecasting the yield of a food production company, while also predicting yield evolution based on financial market values. The methodology involves conducting a survey of the state of the art, analyzing and designing the client/server application, implementing the backend, developing the frontend for mobile platforms, and integrating prediction/estimation models. This platform aims to enhance efficiency and decision-making in food production businesses by providing comprehensive management and forecasting capabilities.

Keywords: Mobile application; backend; Income over Feed Cost; Food production

Resumo

A produção de alimentos é um modelo de negócio que utiliza recursos como espaço, alimento ou outros consumíveis para fornecer sustento, geralmente para humanos. Este trabalho tem como objetivo estudar e desenvolver um sistema de gestão e previsão de produção alimentar, com foco principal na produção de laticínios, permitindo aos utilizadores gerir a produção e aceder a indicadores úteis para as operações da empresa, como o Input IOFC, e prever custos/receitas com base nas tendências do mercado financeiro. O objetivo é desenvolver uma plataforma móvel com um backend acessível via REST/ JSON para gerir e prever o rendimento de uma empresa de produção alimentar, prevendo também a evolução do rendimento com base nos valores do mercado financeiro. A metodologia envolve realizar um levantamento do estado da arte, analisar e projetar a aplicação cliente/servidor, implementar o backend, desenvolver o frontend para plataformas móveis e integrar modelos de previsão/estimativa. Esta plataforma visa melhorar a eficiência e tomada de decisão nas empresas de produção alimentar, fornecendo capacidades abrangentes de gestão e previsão.

Palavras-chave: Aplicação móvel; backend; Income over Feed Cost; Produção alimentar.

Conteúdo

Lista de Figuras

Acrônimo

API Application Programming Interface.

CRUD Create, read, update and delete.

DB Data Base.

ESTiG Escola Superior de Tecnologia e Gestão.

FKMD Fundação Klibur Matadalan.

HTTP Hypertext Transfer Protocol.

HTTPS Hypertext Transfer Protocol Secure.

IA Inteligência Artificial.

IOFC Income Over Feed Cost.

IOS iPhone Operating System.

IPB Instituto Politécnico de Bragança.

JSON Javascript Object Notation.

JWT JSON Web Token.

ORM Object-Relational Mapping.

PDF Portable Document Format.

PHP Personal Home Page.

REST Representational State Transfer.

SQL Structured Query Language.

UML Unified Modeling Language.

URL Uniform Resource Locator.

Capítulo 1

Introdução

Grandes desafios são colocados à humanidade neste século XXI, entre os quais se destaca a necessidade de produzir alimentos para uma população em constante crescimento, sem comprometer os recursos naturais e o meio ambiente. É crucial desenvolver investigações que não só aumentem a produção, mas também criem empregos e rendimento, promovendo o desenvolvimento sustentável das pessoas e das regiões envolvidas. Este esforço não pode ser isolado; deve envolver uma colaboração entre diferentes instituições, áreas de pesquisa e até mesmo países, todos convergindo para objetivos semelhantes. A produção de leite, como qualquer atividade produtiva, deve gerar lucro ao produtor, garantindo a sua continuidade no setor. O lucro da atividade está diretamente ligado à produtividade, aos custos de produção e ao preço do leite pago ao produtor. Reduzir despesas com consumíveis e gerir eficazmente imprevistos, como doenças ou desastres naturais, são estratégias para aumentar o lucro. Para enfrentar esses desafios, a implementação de sistemas de gestão e previsão de produção alimentar torna-se essencial. Estudos recentes demonstram a eficácia da aplicação de técnicas de Inteligência Artificial (IA) na agricultura. Segundo Patra [1] e Kutyauripo et al. [2], diversas tecnologias estão a ser utilizadas para melhorar a previsão de rendimentos e a gestão de recursos agrícolas. Estes estudos incluem exemplos específicos na produção de leite, destacando como aplicativos móveis e inteligência artificial podem ajudar os produtores a otimizar custos, prever receitas e tomar decisões mais informadas.

1.1 Enquadramento

Dado o contexto mencionado anteriormente, aplicação MilkPoint surgiu para explorar uma área em expansão na agricultura, especificamente na produção leiteira. Estas plataformas são especialmente úteis no âmbito empresarial, considerando a necessidade de aplicativos móveis para Android que melhorem a tomada de decisões e prevejam custos e receitas. A implementação de tecnologias avançadas, como a inteligência artificial e a análise de dados, permite aos produtores gerir de forma mais eficiente os seus recursos, aumentar a produtividade e assegurar a sustentabilidade das operações. Com estas ferramentas, é possível não apenas otimizar a gestão diária, mas também antecipar e mitigar imprevistos, como doenças ou condições climáticas adversas, garantindo assim um maior lucro e a continuidade da atividade no setor leiteiro.

1.2 Objetivo

Foi com a ambição de criar o tipo de plataforma mencionada anteriormente que a MilkPoint surgiu. Entre os diversos serviços a serem implementados, destaca-se o desenvolvimento de uma aplicação móvel para Android voltada para o cálculo do IOFC. Esta aplicação permitirá o registo de uma exploração leiteira e a consulta de diversas informações relacionadas, além de previsões baseadas nas tendências históricas do IOFC armazenadas na base de dados. Para alcançar esse objetivo, será utilizada a linguagem Flutter para o frontend da aplicação e Laravel e Personal Home Page (PHP) para o backend, com comunicação via JSON.

São objetivos específicos:

- Criação de uma plataforma móvel, especificamente para o sistema operativo android intuitiva e eficiente para facilitar o acesso e a gestão dos dados de produção alimentar.
- Implementação de indicadores de desempenho, incluindo o cálculo do IOFC, para avaliar a eficiência económica da produção de leite.

- Desenvolvimento de modelos de previsão de custos e receitas de IOFC baseados em dados históricos, proporcionando insights para o planeamento estratégico e financeiro

1.3 Estrutura do Documento

O documento encontra-se estruturado em 6 capítulos, conforme apresentado na Tabela 1.1. Cada um abordando aspetos fundamentais do projeto.

Nome do Capítulo	**Conteúdo**
Introdução	Neste capítulo, é apresentada a introdução ao projeto, incluindo os objetivos e metas a serem alcançados, bem como a justificativa para a realização do trabalho.
Estado da Arte, Abordagem, Tecnologia	Este capítulo aborda a revisão da literatura e a análise de trabalhos semelhantes. Também são descritas as tecnologias utilizadas no projeto e as metodologias e abordagens adotadas.
Análise e Metodologia	Aqui, são definidos os requisitos funcionais e não funcionais do sistema. É apresentada a arquitetura do sistema e a metodologia de desenvolvimento utilizada, destacando-se o modelo cascata.
Desenvolvimento	Este capítulo detalha a implementação do backend, o desenvolvimento do frontend para plataformas móveis e a integração dos modelos de previsão e estimativa.
Teste e Avaliação	Neste capítulo, são apresentados os testes realizados para verificar se o projeto cumpre os objetivos assumidos. Os resultados de cada teste são precedidos por uma descrição resumida do teste e dos resultados esperados. Também são comentados os resultados obtidos, inferências possíveis, aspectos que poderiam ser diferentes, onde se foi além dos objetivos iniciais e os objetivos que ficaram por cumprir e porquê.
Conclusão	O capítulo de conclusão sintetiza e fornece uma visão geral do trabalho desenvolvido. Faz referência a trabalhos semelhantes e aos conhecimentos emergentes, além de apresentar sugestões para trabalhos futuros, mantendo coerência com as ideias principais da introdução.

Tabela 1.1: Estrutura recomendada do Relatório do Projeto.

Capítulo 2

Contexto e Tecnologias

O crescimento constante da população mundial coloca uma pressão significativa sobre a produção de alimentos [3]. Estima-se que, até 2050, a população global atinja cerca de 10 bilhões de pessoas, o que exigirá um aumento substancial na produção de alimentos para atender a esta demanda. Este desafio é exacerbado pela urbanização crescente e pelas mudanças nos padrões alimentares, que tendem a aumentar o consumo de produtos de origem animal, como laticínios, carne e ovos. Portanto, é crucial encontrar formas de aumentar a produção de alimentos de forma eficiente e sustentável. A produção agrícola e pecuária tem um impacto significativo no meio ambiente, contribuindo para a degradação dos solos, a contaminação das águas, a emissão de gases com efeito de estufa e a perda de biodiversidade. A sustentabilidade ambiental na produção de alimentos é essencial para garantir que os recursos naturais sejam preservados para as gerações futuras. Isso implica a adoção de práticas agrícolas sustentáveis, a redução do uso de produtos químicos nocivos, a implementação de sistemas de gestão de resíduos eficientes e a promoção da biodiversidade nas áreas de cultivo. A eficiência na utilização de recursos é um dos principais desafios na produção de alimentos. Isso envolve a otimização do uso de água, energia, fertilizantes e outros insumos agrícolas para maximizar a produtividade sem comprometer a sustentabilidade. A inovação tecnológica desempenha um papel crucial neste aspeto, oferecendo ferramentas e técnicas que permitem monitorizar e gerir recursos de forma mais precisa. A introdução de sistemas de gestão de produção, como a aplicação

móvel descrita neste trabalho, pode ajudar os produtores a tomar decisões informadas, reduzir desperdícios e aumentar a eficiência geral da produção

2.1 Especificidades da Produção de Laticínios

A gestão da produção leiteira é uma tarefa complexa que envolve a coordenação de diversas atividades, desde a alimentação e o cuidado com os animais até a recolha e processamento do leite. A eficiência na gestão destas atividades é crucial para garantir a qualidade e a quantidade do leite produzido. A utilização de tecnologias de monitorização e gestão pode ajudar a otimizar o processo, permitindo um acompanhamento mais preciso da saúde dos animais, da nutrição e da produtividade. A implementação de sistemas de gestão informatizados pode facilitar a organização das rotinas diárias, a monitorização do desempenho e a tomada de decisões informadas. Os custos de produção na indústria de laticínios são influenciados por diversos fatores, incluindo o preço dos alimentos para os animais, os custos de manutenção das instalações, os gastos com medicamentos veterinários e os custos de mão-de-obra. A gestão eficaz dos custos de produção é essencial para garantir a rentabilidade da atividade leiteira. A utilização de ferramentas de gestão que permitam o controlo e a análise detalhada dos custos pode ajudar os produtores a identificar áreas de desperdício, a otimizar o uso de recursos e a reduzir as despesas operacionais. Os preços de mercado do leite são sujeitos a variações que dependem de fatores como a oferta e a procura, as políticas agrícolas, as condições climáticas e as tendências de consumo. Estas variações podem afetar significativamente as margens de lucro dos produtores de leite. Para manter a sustentabilidade económica, é crucial que os produtores sejam capazes de prever e reagir às flutuações do mercado. A integração de modelos preditivos e a análise de dados de mercado podem fornecer informações valiosas para a tomada de decisões estratégicas, permitindo ajustar a produção e a comercialização de forma a maximizar os lucros.

2.2 Aplicações e Abordagens Semelhantes

Estudos e tendências atuais indica a produção de laticínios enfrenta atualmente o desafio de aumentar a eficiência sem comprometer a sustentabilidade ambiental. Estudos recentes indicam que práticas de manejo sustentável, como a rotação de pastagens e a redução do uso de antibióticos, podem melhorar a saúde do solo e reduzir a pegada de carbono da produção leiteira [4]. Desafios e soluções propostas entre os principais desafios enfrentados pela produção de leite estão as emissões de gases de efeito estufa, o uso intensivo de água e a gestão de resíduos. Soluções propostas incluem a adoção de tecnologias de digestão anaeróbio para converter resíduos em biogás, a implementação de sistemas de irrigação eficientes e a utilização de rações otimizadas para reduzir a produção de metano pelos animais [5].

Existem várias ferramentas de software para a gestão da produção de laticínios, como DairyComp 305, DelPro e Uniform-Agri. Estes sistemas ajudam os produtores a monitorizar a saúde do rebanho, a produção de leite e a gestão de recursos, mas apresentam limitações em termos de custo e complexidade de uso [6].

As principais funcionalidades incluem a monitorização da saúde animal, a gestão de reprodução, o controlo de qualidade do leite e a gestão financeira [7]. No entanto, as limitações incluem a necessidade de formação especializada para utilizar os softwares eficazmente e a falta de integração com outras tecnologias agrícolas [8].

As aplicações móveis na agricultura oferecem várias vantagens, como o acesso em tempo real a dados de campo, a facilidade de uso e a redução de custos operacionais. No entanto, também apresentam desvantagens, incluindo a dependência de uma boa conexão à Internet e a necessidade de dispositivos móveis robustos para suportar condições adversas no campo [9] .

Exemplos de aplicações móveis bem-sucedidas como iCow, uma aplicação que fornece informações sobre cuidados com o gado e gestão de lactação para agricultores em África [10]

A previsão de custos e receitas é essencial na gestão da produção agrícola, incluindo

a produção de leite. O uso de modelos preditivos, especialmente o forecasting de séries temporais, tem se mostrado eficaz para prever valores futuros com base em dados históricos.

O forecasting de séries temporais é uma técnica que analisa dados históricos para identificar padrões e tendências futuras. Métodos comuns incluem regressão linear, árvores de decisão e redes neurais. Esta técnica é particularmente útil para prever variações nos custos e receitas ao longo do tempo [11].

Um estudo de caso analisa a implementação de uma aplicação móvel AI-driven em uma fazenda leiteira nos EUA. A aplicação foi utilizada para monitorizar a saúde das vacas, otimizar a alimentação e prever custos e receitas [12].

Os sistemas de gestão de produção alimentar são ferramentas essenciais que permitem aos produtores monitorizar, controlar e otimizar todas as etapas do processo produtivo [13]. Através da coleta e análise de dados, esses sistemas ajudam a aumentar a eficiência, reduzir desperdícios e melhorar a qualidade dos produtos. A importância de um sistema de gestão bem implementado reside na sua capacidade de integrar diversas operações, proporcionando uma visão abrangente e detalhada da produção, o que é crucial para a tomada de decisões estratégicas e operacionais.

O Modelo de negócio é o coração da aplicação, desempenhando um papel fundamental na lógica de processamento e manipulação de dados. Ela contém as regras que governam o funcionamento do sistema, assegurando que as operações realizadas sejam consistentes e corretas. Neste projeto foram definido as entidade como `User` representa os utilizadores que interagem com a aplicação, armazenando informações como nome, email e dados de autenticação. `Stock` refere-se ao controlo dos estoques de leite, incluindo atributos como quantidade de leite produzido, data de registo e número das vacas responsáveis pela produção. `Revenue e Expense` estas entidades são essenciais para a gestão financeira, representando, respetivamente, as receitas e despesas. Elas incluem dados como montante, data e tipo de transação (por exemplo, venda ou compra).

As regras de negócio são fundamentais para definir como os dados podem ser manipulados e processados dentro da aplicação. Estes foram definidos no projeto como `Cálculo`

`de Receitas Diárias` implementação de lógica para calcular o total de receitas geradas diariamente. `Cálculo de Receitas Mensais` implementação de lógica para calcular o total de receitas geradas em um determinado mês. `Cálculo de Despesas Diárias` implementação de lógica para calcular o total de despesas incorridas diariamente. `Cálculo de Despesas Mensais` implementação de lógica para calcular o total de despesas incorridas em um determinado mês.`Cálculo do IOFC Diário` implementação de lógica para calcular o IOFC diariamente, subtraindo as despesas das receitas do dia.`Cálculo do IOFC Mensal` implementação de lógica para calcular IOFC mensalmente, subtraindo as despesas das receitas do mês. `Criação de Relatórios` lógica que permite compilar dados armazenados para produzir relatórios financeiros, que podem ser utilizados para tomada de decisão e planeamento. `Criação de Previsão` Implementação de lógica para gerar previsões com base nos custos históricos do IOFC. Esta regra utiliza dados históricos de receitas e despesas para prever o desempenho financeiro futuro, ajudando na tomada de decisões estratégicas.

A camada de visualização é responsável pela interface gráfica da aplicação, onde os utilizadores interagem diretamente com o sistema. Esta camada transforma os dados processados pelo modelo de negócio em informações visíveis e interativas. Os componentes desta camada incluem a Interfaces de Utilizador são elementos visuais que apresentam informações ao utilizador, como relatórios em Portable Document Format (PDF) que podem ser gerados a partir dos dados e as Interações do Utilizador são as formas como os utilizadores interagem com a aplicação, incluindo botões para adicionar ou remover itens.

Camada de Controle atua como um intermediário entre a camada de visualização e o modelo de negócio. Esta camada processa as entradas dos utilizadores, executa a lógica do modelo de negócio e retorna os resultados para a camada de visualização. Para isso, no projeto existe os controladores como `usercontroller`, `stockcontroller`, `revenuecontroller`, `reportcontroller`, `expensecontroller`, `anallysticcontroller` recebem as solicitações dos utilizadores, manipulam os dados através do modelo de negócio e retornam as respostas apropriadas. Estes controladores são responsáveis por coordenar a comunicação entre a camada de visualização e o modelo de negócio, garantindo que os

dados fluam corretamente entre ambas.

Exemplo prático no controlador, um controlador pode ser responsável por operações como a adição de um novo registo de receita. O controlador receberá a entrada do utilizador (como o valor da receita, a data, etc.), validará a entrada, passará os dados ao modelo de negócio para processamento e, em seguida, retornará a confirmação à camada de visualização. Na rotas configura uma rota para adicionar uma nova receita para chamar um método específico no controlador de receitas.

No fluxo de dados quando um utilizador interage com a interface (por exemplo, adicionando uma nova receita), a solicitação é enviada ao controlador correspondente. O controlador processa esta solicitação, utiliza os modelos de negócio para atualizar os dados e, finalmente, retorna uma resposta à interface do utilizador. Essa estrutura são muito importante na camada de controle como diretórios `controllers/` contém os controladores que lidam com as solicitações e coordenam a lógica do negócio.`'models/'` contém os modelos de dados e a lógica do negócio. `'views/'` contém os componentes de interface do utilizador. Camada de Application Programming Interface (API)/Comunicação é essencial em uma aplicação móvel como o MilkPoint, especialmente quando a aplicação precisa se comunicar com um servidor para acessar e manipular dados. Esta camada lida com a troca de dados entre a aplicação cliente (no dispositivo móvel) e o servidor backend. Ela utiliza API para facilitar essa comunicação de maneira eficiente e segura.Componentes desta camada que é o Endpoints de API que tem como permitem a realização de operações Create, read, update and delete (CRUD) em recursos como despesas, receitas, estoques e usuários. Exemplo de Um endpoint para listar todas as despesas pode ser definido como `GET /api/expenses` , enquanto um endpoint para criar uma nova despesa pode ser `POST /api/expenses`, e um endpoint para eliminar uma despesa pode ser `DELETE /api/expenses/id`. O outro componente que é o Protocolo Hypertext Transfer Protocol (HTTP)/Hypertext Transfer Protocol Secure (HTTPS). O HTTP e HTTPS são os protocolos utilizados para comunicação entre a aplicação cliente e o servidor. tem como função permite a transferência segura de dados. O uso de HTTPS é especialmente importante para proteger dados sensíveis durante a transmissão. Neste projeto a camada

de API/Comunicação é implementada usando o framework Laravel para o backend. Os controladores no Laravel são responsáveis por definir os endpoints da API, manipulando as requisições e respondendo com os dados apropriados.

A camada de persistência de dados é fundamental em qualquer aplicação que precise armazenar e recuperar informações. Ela gere como os dados são armazenados, organizados e acedidos, tanto em bases de dados locais quanto remotas. No projeto MilkPoint, esta camada é responsável por garantir que todas as informações sobre despesas, receitas, stocks e utilizadores sejam guardadas e recuperadas de forma eficiente e segura. Os componentes desta camada incluem a base de dados Structured Query Language (SQL) e o mecanismo de cache.

A base de dados SQL tem como função armazenar dados de maneira organizada, permitindo operações de CRUD. Exemplos de bases de dados SQL são MySQL e PostgreSQL. Por outro lado, o mecanismo de cache consiste em sistemas que armazenam temporariamente dados frequentemente acedidos para melhorar a performance. A função do cache é reduzir o tempo de resposta ao armazenar cópias de dados frequentemente solicitados.

Os componentes principais incluem:

- Frontend: A interface do utilizador, desenvolvida para dispositivos móveis com sistema operativo Android, permitirá aos produtores inserir dados, visualizar relatórios e monitorizar o desempenho em tempo real [14]. O frontend será construído utilizando a framework Flutter, conhecida pela sua capacidade de criar interfaces de utilizador nativas e de alta performance .

- Backend: [15]Responsável pelo processamento dos dados e pela lógica de negócios, o backend será implementado em Laravel, um framework PHP robusto e escalável. O backend irá gerir a comunicação entre o frontend e a base de dados, garantindo que as operações de criação, leitura, atualização e exclusão CRUD sejam realizadas de forma segura e eficiente .

- Bases de dados: A base de dados armazenará todas as informações relacionadas à

produção, incluindo custos, receitas, despesas e indicadores de desempenho. Utilizando um sistema de gestão de bases de dados relacional (como MySQL ou PostgreSQL), os dados serão organizados de maneira a facilitar a sua recuperação e análise.

2.3 Indicadores de Desempenho na Produção de Laticínios

IOFC é um indicador crucial para a gestão da produção de laticínios, pois mede a eficiência económica do uso de alimentos em relação à produção de leite. O cálculo do IOFC envolve a subtração dos custos de alimentação dos rendimentos obtidos com a venda do leite, proporcionando uma medida clara da rentabilidade. Este indicador permite aos produtores avaliar o impacto das decisões de alimentação na lucratividade e ajustar as estratégias conforme necessário. Além do IOFC, outros indicadores de desempenho podem ser utilizados para fornecer uma visão mais completa da produção de laticínios, tais como: Produção média de leite por vaca, Custo por litro de leite produzido. Previsão de Custos e Rendimentos com Base na Evolução do Mercado Financeiro é outro indicador relevante a previsão de custos e receitas com base na evolução do mercado financeiro. Este indicador utiliza modelos de previsão para analisar tendências de preços de insumos e produtos finais no mercado financeiro, ajudando os produtores a antecipar variações nos custos de produção e nos preços de venda do leite. A análise pode incluir fatores como tendências de mercado para produtos lácteos que é a análise de tendências de preços para produtos lácteos no mercado, ajudando a prever receitas futuras.

2.4 Ferramentas e Justificação para seu Uso

A escolha das ferramentas para o desenvolvimento do sistema de gestão de produção alimentar foi baseada em critérios como eficiência, escalabilidade. A eficiência refere-se à capacidade das ferramentas em realizar tarefas rapidamente e com alta performance.

A escalabilidade é crucial para garantir que o sistema possa crescer e adaptar-se a um número crescente de utilizadores e dados sem perda de desempenho.

Flutter foi escolhido para o desenvolvimento do frontend devido à sua capacidade de criar interfaces de utilizador nativas e de alta performance, especificamente para o sistema operativo Android. Flutter permite um desenvolvimento rápido e eficiente com uma única base de código, facilitando a manutenção e a implementação de novas funcionalidades. Laravel foi a ferramenta selecionada para o desenvolvimento do backend. Este framework PHP é conhecido pela sua robustez, facilidade de uso e uma vasta gama de funcionalidades integradas que facilitam o desenvolvimento de aplicações web seguras e escaláveis. Laravel também oferece um suporte excelente para a gestão de bases de dados, autenticação de utilizadores e criação de API. Para a comunicação entre o frontend e o backend, foram utilizadas API REST/JSON. Estas API permitem uma troca de dados eficiente e estruturada entre os diferentes componentes do sistema, garantindo que o frontend possa enviar e receber informações de forma rápida e segura. A utilização de API REST/JSON facilita a integração e a escalabilidade do sistema. Para a comunicação entre o frontend e o backend, foram utilizadas API REST/JSON. Estas API permitem uma troca de dados eficiente e estruturada entre os diferentes componentes do sistema, garantindo que o frontend possa enviar e receber informações de forma rápida e segura. A utilização de API REST/JSON facilita a integração e a escalabilidade do sistema.

2.4.1 Frontend

Flutter foi escolhido para o desenvolvimento do frontend devido à sua capacidade de criar interfaces de utilizador nativas e de alta performance para o sistema operativo Android.O frontend oferece várias funcionalidades cruciais para a gestão da produção de laticínios, incluindo:

- Criação de despesas: Permite aos utilizadores registar todas as despesas relacionadas com a produção de laticínios se é comprada ou produzida.
- Criação de receitas: Permite aos utilizadores registar todas as receitas provenientes

da venda de leite e produtos derivados.

- Gestão de stock de leite. Permite monitorizar e gerir os níveis de stock de leite disponível.

- Nova venda: Permite aos utilizadores registar novas vendas, incluindo a quantidade de litros vendidos, o valor da venda e a data da transação.

- Relatório: Fornece ferramentas para gerir relatórios, incluindo IOFC diário e mensal, despesas, receitas e previsões.

Para a comunicação com o backend, é utilizado o formato JSON. Este formato facilita a troca de dados estruturados e a integração entre diferentes componentes do sistema.

2.4.2 Backend

Laravel foi escolhido como framework de desenvolvimento do backend devido à sua robustez, escalabilidade e suporte a práticas de desenvolvimento modernas. O backend é responsável por várias funcionalidades essenciais:

- Gestão de Base de Dados: Inclui operações CRUD para todas as entidades do sistema.

- Processamento de requests: Trata e processa todas as requisições recebidas do frontend, garantindo que os dados sejam manipulados corretamente.

- Gerir as previsões: Utiliza modelos de machine learning para prever custos e receitas futuros, com base em dados históricos.

A comunicação entre o frontend e o backend é realizada através de API REST/JSON, permitindo uma integração eficiente e segura entre os dois componentes.

2.4.3 Base de Dados

A base de dados é estruturada para suportar todas as funcionalidades do sistema, incluindo o armazenamento de dados de despesas, receitas, estoque de leite e previsões. A modelagem da base de dados é realizada de forma a garantir a integridade e a consistência dos dados. Laravel facilita a interação com a base de dados através do seu Object-Relational Mapping (ORM). Isto permite que os dados sejam persistidos de forma eficiente e segura, utilizando uma sintaxe intuitiva e fácil de utilizar.

Capítulo 3

Análise e Metodologia

A gestão da produção de leite depende de informação rigorosa e atualizada. A maximização do lucro e aumento da qualidade da produção requer a análise constante de vários parâmetros que incluem, entre outros, as despesas com a alimentação e tratamento dos animais, quantidade de leite produzido, preço dos consumíveis, entre outros. Um dos indicadores mais importantes é o IOFC e a previsão dos custos/rendimentos com base na evolução do mercado financeiro, neste caso no projeto com base no valores IOFC e dados históricos. Este mede a produtividade, ou seja, o valor que fica após o custo com a alimentação das vacas leiteiras. Este é um dado crítico aquando da avaliação económica de uma exploração leiteira, sendo calculado com base nos seguintes dados: preço médio pago do leite por mês, leite total produzido/dia/mês, número de vacas a produzir no mês, alimentação utilizada, custo por tonelada consumida (produzidas e compradas).

3.1 Arquitetura do Sistema

A arquitetura do sistema é composta por três componentes principais: frontend, backend e base de dados. A comunicação entre o frontend e o backend é realizada utilizando API REST/JSON, garantindo a troca eficiente de informações. Seguinte são a Comunicação entre frontend e backend, como na Figura 3.1.

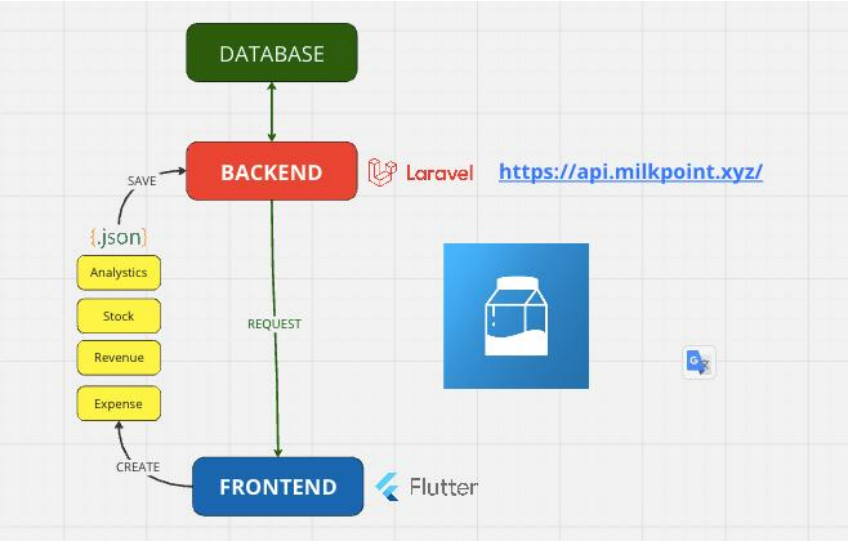


Figura 3.1: Estrutura da Aplicação.

- O frontend, desenvolvido em Flutter, comunica-se com o backend através de chamadas API. Estas chamadas são realizadas utilizando o formato JSON, permitindo uma estrutura de dados legível e fácil de manipular. A comunicação entre o frontend e o backend pode ser tanto bidirecional quanto unidirecional, dependendo da natureza da operação:

  - Estrutura bidirecional de requests: Utilizada para operações que requerem feedback imediato do servidor, como a criação de novas entradas (despesas, receitas) ou a atualização de dados existentes.
  - Estrutura unidirecional de requests: Utilizada para operações onde o frontend apenas envia dados ao backend sem necessitar de uma resposta imediata, como registos de logs ou notificações.

3.2 História de Utilizador

Para o projeto foi definida os requisitos do sistema, como ilustrado pela Tabela 3.1.

A plataforma MilkPoint foi desenvolvida para facilitar a gestão e previsão da produção de laticínios, permitindo que os produtores registrem e visualizem vendas, estoques, e

Código	História de Utilizador
C001	Como produtor, gostaria de criar uma conta
C002	Como produtor, gostaria de registar uma nova venda de leite
C003	Como produtor, gostaria de visualizar a venda registada
C004	Como produtor, gostaria de registar uma nova produção ou Estoque de leite
C005	Como produtor, gostaria de visualizar o Estoque registada
C006	Como produtor, gostaria de registar um novo custo
C007	Como produtor, gostaria de visualizar os custos introduzidos
C008	Como produtor, gostaria de criar o relatório diário de IOFC
C009	Como produtor, gostaria de criar o relatório mensal de IOFC
C010	Como produtor, gostaria de criar o relatório diário de receitas
C011	Como produtor, gostaria de criar o relatório mensal de receitas
C012	Como produtor, gostaria de criar o relatório diário de despesa
C013	Como produtor, gostaria de criar o relatório mensal de despesa
C014	Como produtor, gostaria de visualizar a previsão por dia
C015	Como produtor, gostaria de visualizar a previsão por semana
C016	Como produtor, gostaria de visualizar a previsão por mês

Tabela 3.1: História de Utilizador

despesas de forma simples e eficiente. Além disso, a aplicação oferece a criação de relatórios diários e mensais de IOFC, ajudando na análise da rentabilidade e na otimização das operações. Com funcionalidades de registo detalhado e geração de relatórios, a plataforma apoia os produtores na tomada de decisões informadas, visando uma gestão mais eficaz e sustentável da produção.

3.3 Caso de Uso

O uso de diagramas Unified Modeling Language (UML) é extremamente útil para visualizar graficamente os detalhes do sistema e como os utilizadores podem interagir com o mesmo. Neste contexto, foi desenvolvido um diagrama de casos de uso que mostra a relação entre os utilizadores do sistema e as funcionalidades disponíveis. Este diagrama proporciona uma representação clara das ações que os utilizadores podem realizar na aplicação, facilitando a compreensão das interações e dos fluxos de trabalho dentro do sistema de gestão de produção alimentar. A imagem seguinte ilustra 3.2 o diagrama do projeto.

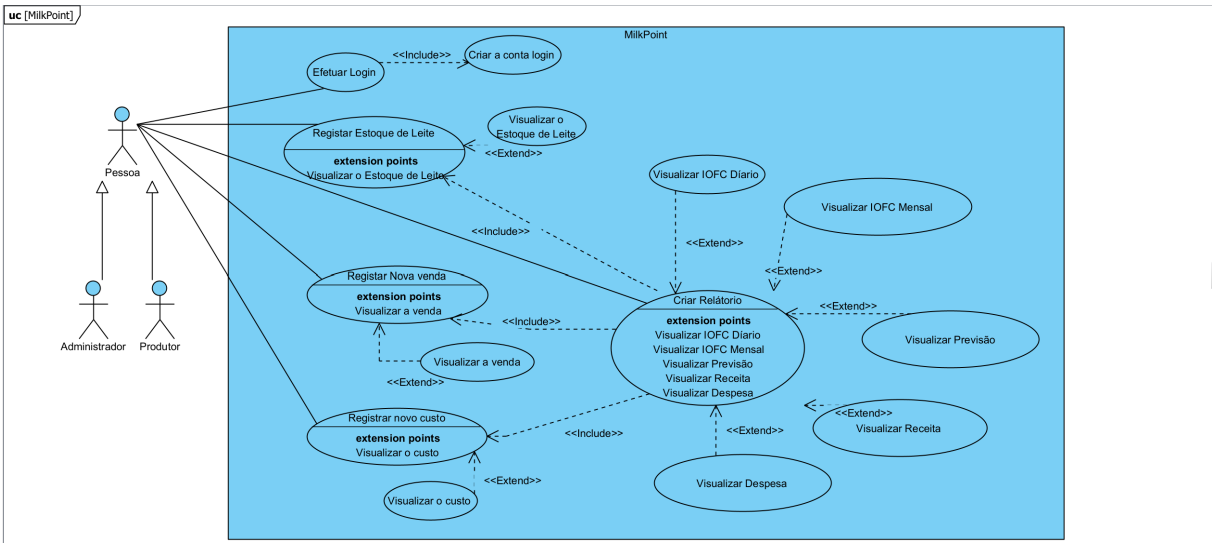


Figura 3.2: Caso de uso da aplicação.

3.4 Requisitos da Aplicação

No contexto do projeto de plataforma de gestão e previsão de produção alimentar, os requisitos podem ser divididos em requisitos funcionais e requisitos não funcionais. Os requisitos funcionais são:

- Autenticação e conta de usuário: O sistema deve permitir que os produtores criem uma conta utilizando nome, email, telefone e senha.
- Gestão de vendas: O sistema deve permitir o registo de vendas de leite, incluindo a quantidade de leite vendida e a data da transação. Os produtores devem poder visualizar as vendas registadas no sistema.
- Gestão de estoque: Os produtores devem poder registar a produção de leite, incluindo a quantidade produzida e a data da produção. Os produtores devem poder visualizar o estoque atual de leite disponível.
- Registo de custos:Os produtores devem poder registar os custos produzida ou comprada relacionados com a produção de laticínios. Os produtores devem poder visualizar os custos registados no sistema. Os produtores devem poder visualizar os custos registados no sistema.

- Relatórios : O sistema deve gerar relatórios diários de IOFC, mensais, de receitas e despesas.

- Previsão com base em dados históricos: O sistema deve utilizar modelos de previsão baseados em dados históricos de IOFC para prever tendências futuras.

Além dos requisitos funcionais, foram considerados requisitos não funcionais para esta aplicação. Estes requisitos não funcionais incluem desempenho, segurança, usabilidade e manutenção:

- Desempenho: O sistema deve ser capaz de suportar um grande volume de transações de dados sem degradação significativa de desempenho.

- Segurança: Os dados dos produtores devem ser armazenados de forma segura e protegidos contra acesso não autorizado.

- Usabilidade: A interface do sistema deve ser intuitiva e fácil de usar para os produtores, independentemente do dispositivo utilizado.

- Manutenção: O sistema deve ser facilmente mantido e atualizado para incorporar novos requisitos e correções de bugs.

3.5 Entidades da Aplicação

Este projeto é voltado para a gestão de uma exploração leiteira, onde são registadas as atividades relacionadas com a produção, venda e despesas envolvendo leite. Utilizadores podem realizar operações como monitoramento de estoque de leite, registo de receitas provenientes da venda de leite, e gestão de despesas associadas à exploração. A Figura 3.3 mostra cada tabela com os seus respetivos atributos e tipos de dados.

- Tabela `Utilizador`: Armazena informações dos utilizadores que acessam o sistema de gestão.

- Tabela `Estoque`: Regista o inventário de leite, incluindo a quantidade produzida e as vacas associadas.
- Tabela `Receita`: Regista as receitas geradas pela venda de leite, incluindo quantidade vendida e montante financeiro.
- Tabela `Despesas`: Regista as despesas operacionais da exploração leiteira, como compras de leite e outros custos associados.

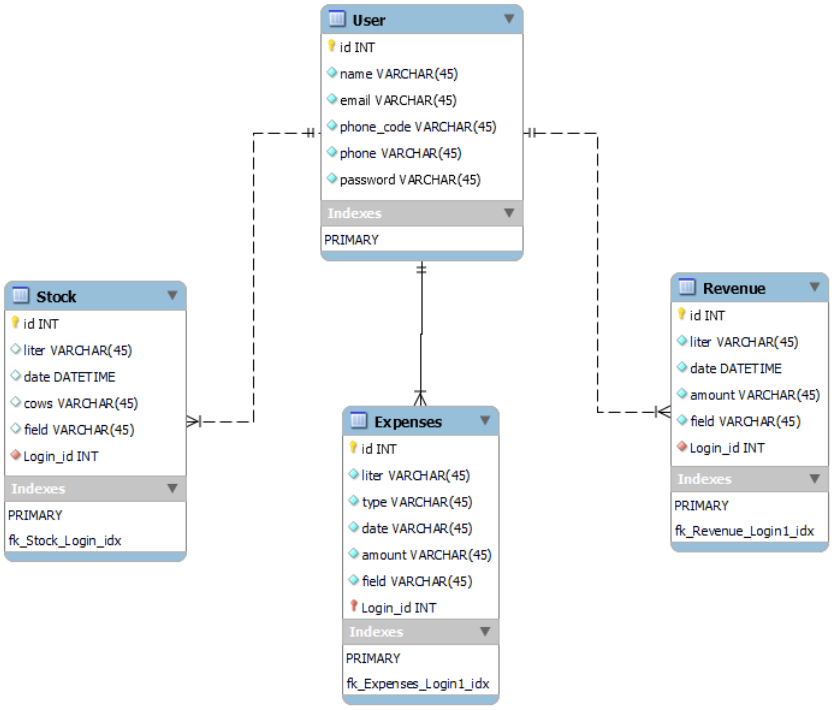


Figura 3.3: Entidades da Aplicação.

3.6 Criação Interface Mockup

A Figura 3.4 o trabalho realizado no Figma, uma ferramenta de design colaborativo que foi utilizada para criar os mockups da aplicação. Os mockups incluem diferentes ecrãs da aplicação, como o registo de despesas e receitas, a monitorização do stock de leite, a geração de relatórios de IOFC e as previsões de custos e receitas.

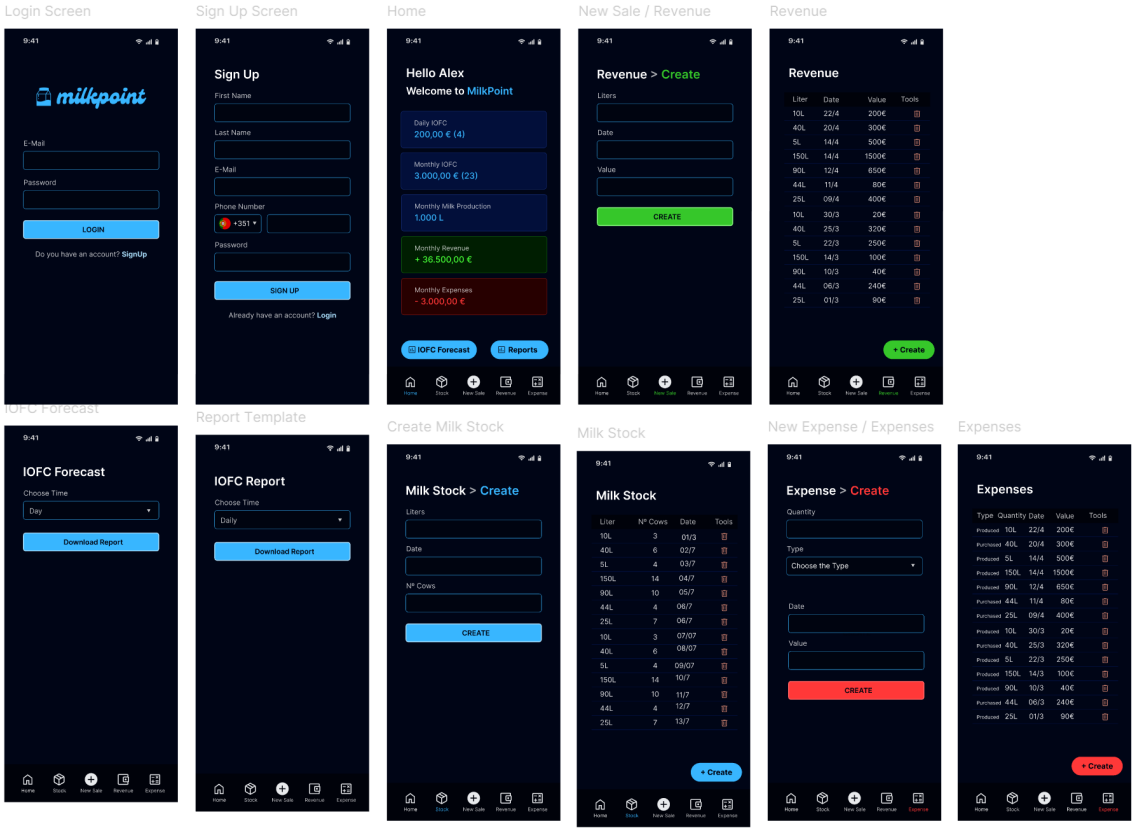


Figura 3.4: Mockup no Figma.

A Figura 3.5 mostra as telas criadas para o utilizador efetuar login na aplicação. Caso o utilizador já possua uma conta, ele poderá ser redirecionado para a página inicial após fazer login. Caso contrário, ele pode criar uma conta e, depois de inserir os dados, será finalmente redirecionado para a página inicial. Na página inicial, o utilizador pode visualizar o IOFC Diário, o IOFC Mensal, a Produção de Leite por Mês, a Receita Mensal e a Despesa Mensal. Além disso, ao clicar no botão ”Relatório”, será direcionado para a página correspondente.

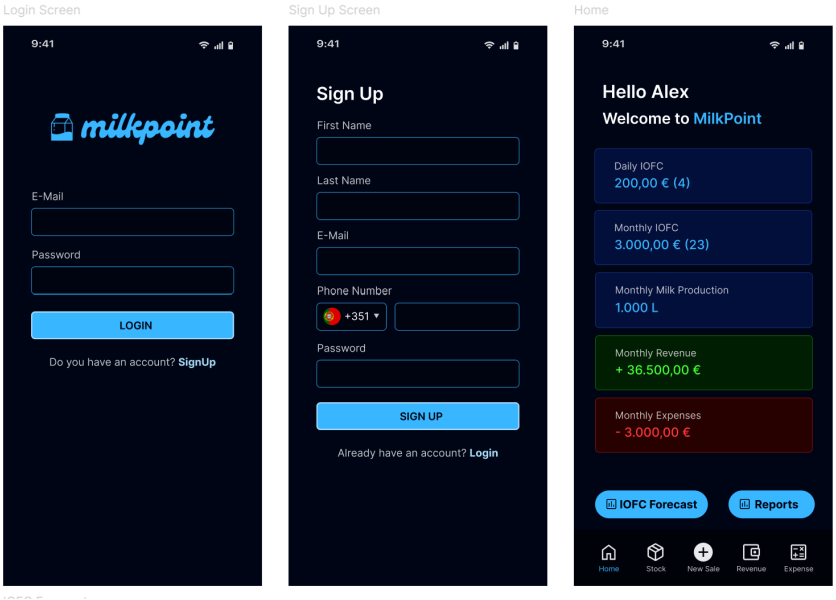


Figura 3.5: Mockup Tela Inicial.

A Figura 3.6 são apresentadas as telas para adicionar nova produção de leite, ou seja, o estoque de leite pode ser atualizado. Após a inserção da nova produção de leite, os utilizadores podem consultar todas as produções realizadas e têm a opção de eliminar aquelas que foram inseridas anteriormente. Esta funcionalidade permite uma gestão eficiente do estoque de leite, garantindo que os dados estejam sempre atualizados e precisos para as operações da empresa de produção de laticínios.

A Figura 3.7 mostra a funcionalidade de nova venda, representa a receita. Os utilizadores podem introduzir dados como a quantidade vendida por litro, o dia da venda e o valor correspondente. Após a inserção dessas informações na base de dados, as vendas

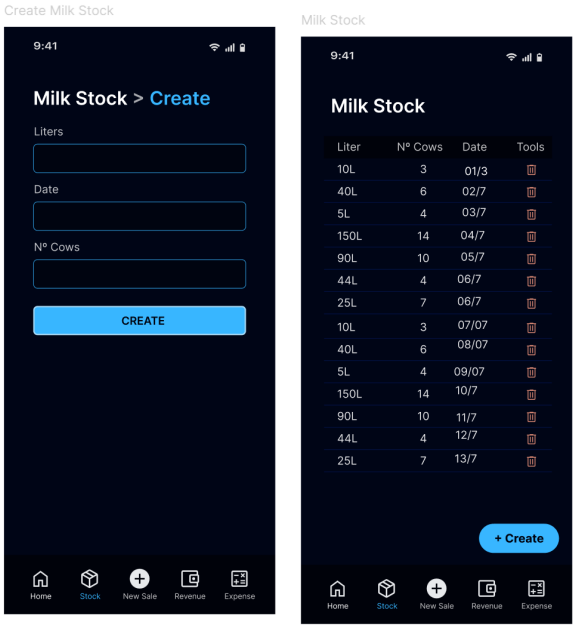


Figura 3.6: Mockup Estoque de Leite.

realizadas podem ser consultadas, visualizadas e também removidas, se necessário. Essa funcionalidade permite um controle detalhado das transações de venda de leite, garantindo que os registos sejam precisos e facilmente geridos pelos utilizadores da aplicação.

A Figura 3.8 mostra a funcionalidade de novo custo , representa a despesa. Os utilizadores podem introduzir novos custos escolhendo o tipo, se são produzidos ou comprados, a quantidade, o dia e o valor gasto. Após a inserção dos novos custos na base de dados, estes podem ser consultados e removidos conforme necessário. Esta funcionalidade permite uma gestão detalhada dos custos associados à produção de laticínios, garantindo que os registos sejam precisos e facilmente geridos pelos utilizadores da aplicação.

A Figura 3.9 mostra a funcionalidade de previsão de IOFC, permitindo aos utilizadores aceder a previsões de desempenho financeiro. Os utilizadores podem escolher o período da previsão, selecionando entre dia, semana ou mês, para obter uma visão detalhada dos dados de acordo com a sua necessidade. Após a geração da previsão, é possível descarregar o relatório para consulta offline. Esta funcionalidade possibilita uma gestão aprofundada

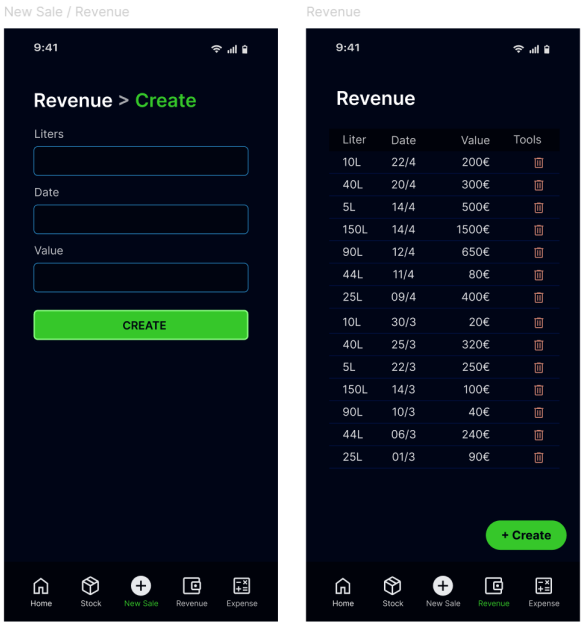


Figura 3.7: Nova Venda.

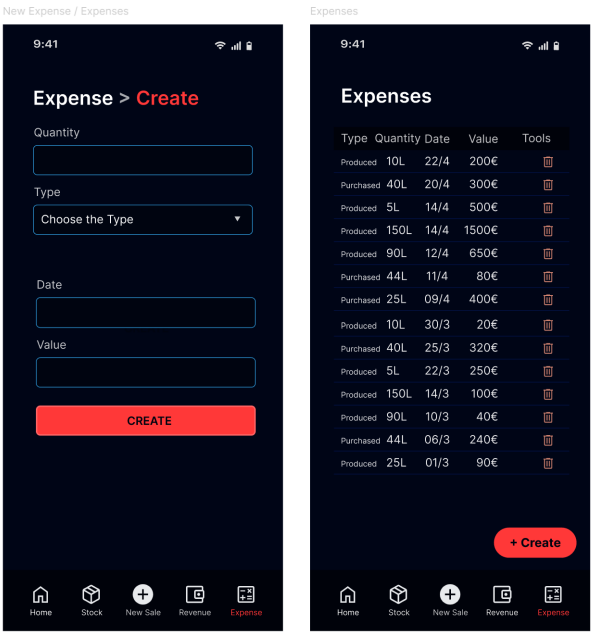


Figura 3.8: Nova Despesa.

e planeamento financeiro, garantindo que os utilizadores possam acompanhar e analisar de forma precisa os dados de previsão de desempenho na aplicação.

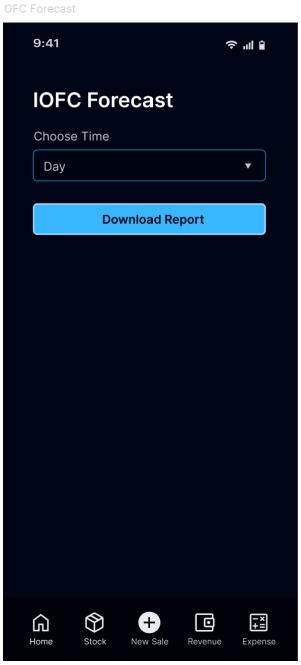


Figura 3.9: Tela de Previsão.

3.7 Metodologia ao Plano de Trabalho

O desenvolvimento da plataforma de gestão e previsão de produção alimentar adotou o modelo cascata, uma abordagem sequencial conhecida por suas fases distintas e lineares, cada uma delas dependendo da conclusão da fase anterior. A Figura 3.10 ilustra a metodologia ou plano de trabalho implementado utilizando o modelo cascata.

A sua vantagem é a estrutura clara e linear que facilita o planeamento e gestão de projetos. Fases bem definidas que permitem um controlo rigoroso sobre o progresso do projeto. Requisitos definidos desde o início, o que facilita a estimativa de custos e prazos. Desvantagem é rigidez na adaptação a mudanças nos requisitos durante o desenvolvimento. Possibilidade de surgirem problemas apenas nas fases finais, quando as mudanças são mais dispendiosas. Menor flexibilidade em comparação com metodologias ágeis para

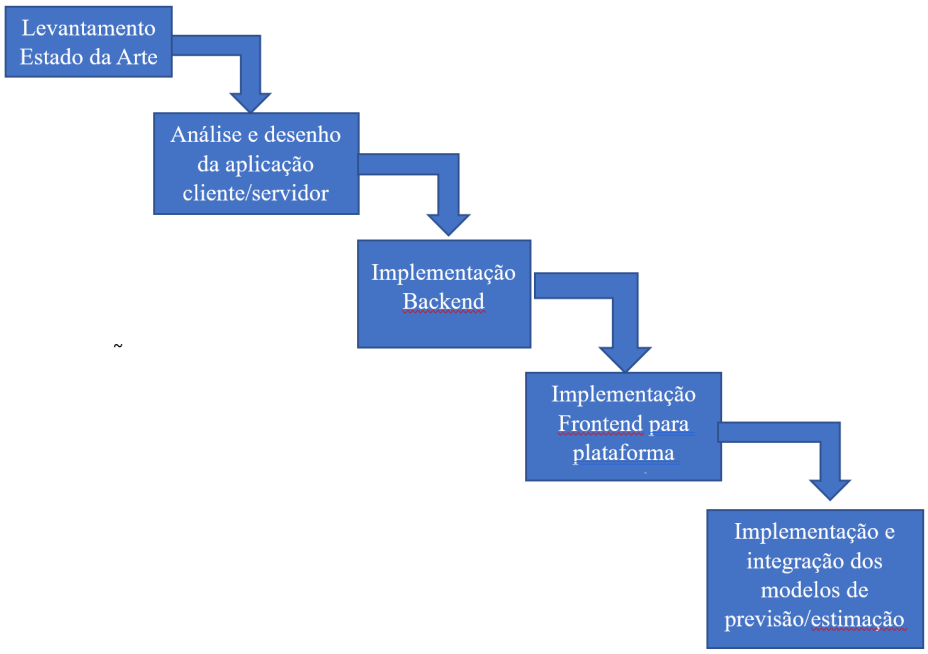


Figura 3.10: Metodologia/Plano de Trabalho.

responder a feedbacks e novas necessidades dos utilizadores.

3.8 Metodologia Implementado ao Desenvolvimento da Aplicação Móvel

Nesta parte também foi implementado o desenvolvimento da plataforma de gestão e previsão de produção alimentar com o modelo cascata.A Figura 3.11 ilustra a metodologia implementado ao desenvolvimento da aplicação.

- Requisitos : Foram definidos os requisitos funcionais e não funcionais do sistema, Também foram definidos história de utilizador para entender melhor as necessidades dos utilizadores.

- Análise: Foi feita a definição da arquitetura ou estrutura do sistema, especificando os componentes e a sua interligação.

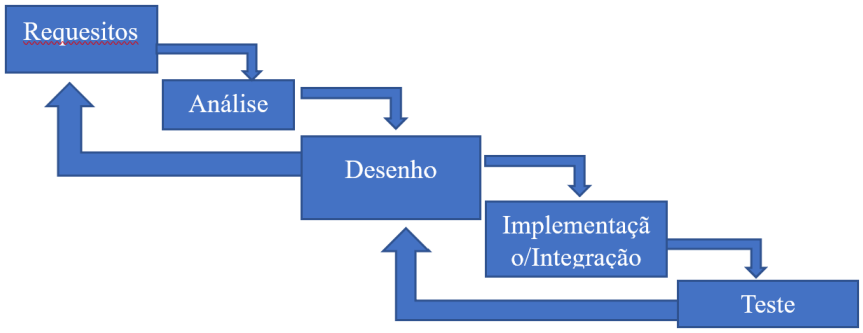


Figura 3.11: Metodologia Implementado ao Desenvolvimento da Aplicação Móvel

- Desenho: Foi desenhada a interação entre a aplicação móvel (cliente) e o servidor. Utilização do Figma para o desenho das interfaces e protótipos da aplicação.
- Implementação e Integração: Implementação do backend utilizando a framework Laravel. Implementação do frontend para plataformas móveis utilizando Flutter.
- Integração dos Modelos de Previsão/Estimação: Implementação e integração dos modelos de previsão com base em dados históricos de IOFC, utilizando Laravel.
- Testes: Realização de testes para verificar se o projeto desenvolvido cumpre os objetivos definidos e resolve o problema descrito na análise. Cada teste é acompanhado por uma descrição do teste realizado e dos resultados esperados, seguidos de uma avaliação dos resultados obtidos.

3.9 Avaliação de Interface

A interface de uma plataforma é o ponto de contacto mais direto entre o usuário e o sistema, sendo essencial para determinar a eficácia de uma aplicação em cumprir seus objetivos. No caso da Plataforma de Gestão e Previsão de Produção Alimentar, a interface desempenha um papel fundamental, já que a eficiência e a clareza das interações podem impactar diretamente a experiência do usuário e a produtividade do processo de

gestão. Assim, a avaliação da interface da plataforma foi concebida com base em critérios amplamente reconhecidos em análises de usabilidade e design de sistemas. A ideia é garantir que o sistema ofereça uma interação intuitiva, visualmente agradável e eficiente, permitindo que os usuários realizem suas tarefas de forma fluida e sem obstáculos. Ao avaliar a interface, é importante considerar não apenas a facilidade de uso, mas também outros fatores que afetam diretamente a experiência do usuário, como a estética visual, a organização dos elementos, a acessibilidade e a resposta às necessidades específicas dos usuários. O objetivo principal dessa análise é garantir que a plataforma seja intuitiva, funcional e visualmente coerente, proporcionando uma experiência satisfatória. Para isso, a avaliação foi realizada sob três grandes pilares: usabilidade, design visual e eficiência funcional. A usabilidade está diretamente relacionada à forma como o usuário interage com o sistema, ou seja, se a interface é fácil de entender e se as funções são acessíveis. A usabilidade de uma plataforma pode ser medida pela facilidade com que os usuários conseguem encontrar e usar as funções disponíveis. Um dos principais desafios ao desenvolver uma interface é equilibrar a simplicidade com a funcionalidade. Uma interface simples demais pode não fornecer todas as opções que o usuário precisa, enquanto uma interface sobrecarregada de informações pode gerar confusão. No caso desta plataforma, a usabilidade foi planeada para que o usuário consiga acessar as principais funcionalidades sem a necessidade de percorrer muitos menus ou submenus. Dessa forma, a estrutura de navegação foi concebida de forma a garantir que o fluxo entre as páginas e funções seja intuitivo. A experiência de navegação entre as diferentes telas precisa ser fluida, sem exigir que o usuário retorne ao menu principal para cada nova ação. Esse é um ponto chave em sistemas de gestão, onde a eficiência do fluxo de trabalho é primordial. Ao avaliar a usabilidade, também é importante observar o número de interações que o usuário precisa realizar para completar uma tarefa. Quanto menos cliques forem necessários, maior será a eficiência da interface. O segundo critério de análise foi o design visual, que abrange todos os aspetos relacionados à estética da plataforma. O design não pode ser apenas agradável aos olhos; ele precisa comunicar as funções da forma mais clara possível. A escolha de cores, por exemplo, desempenha um papel importante nesse contexto. Cores

muito fortes podem causar cansaço visual, enquanto cores suaves demais podem dificultar a identificação de áreas importantes. Para essa plataforma, foi adotado um esquema de cores neutras, buscando um equilíbrio entre uma interface visualmente agradável e funcional. Além disso, a escolha da tipografia também é um fator determinante. Uma fonte bem escolhida, com o tamanho e espaçamento adequados, facilita a leitura e a compreensão dos conteúdos apresentados. A legibilidade do texto é crucial, especialmente em uma aplicação de gestão, onde muitos dados e informações são exibidos. Em termos de organização visual, a disposição dos elementos na tela deve seguir uma lógica clara. Os usuários devem ser capazes de identificar rapidamente os diferentes tipos de informações, como botões de ação, menus, e dados importantes. A hierarquia visual desempenha um papel central nesse processo, ajudando os usuários a distinguir as ações prioritárias das secundárias. Nesse sentido, a avaliação da interface focou em analisar como os elementos estão dispostos e se a interface como um todo apresenta uma estrutura visual coerente, facilitando a navegação e a compreensão das informações. Outro ponto importante no design visual é a responsividade da plataforma, ou seja, sua capacidade de se adaptar a diferentes tamanhos de tela e dispositivos. A acessibilidade não se limita apenas ao design visual, mas também à capacidade da plataforma de ser utilizada por uma ampla gama de usuários, independentemente das limitações que possam ter. Em termos de acessibilidade, uma interface deve ser projetada para permitir que qualquer pessoa, independentemente de suas habilidades físicas ou cognitivas, possa interagir com a plataforma de maneira eficiente. Isso inclui a possibilidade de ajustar o contraste das cores, aumentar o tamanho das fontes, ou até mesmo navegar através de teclas de atalho, em vez de depender exclusivamente do uso de um rato ou toque em telas. Na avaliação da acessibilidade desta plataforma, também foram considerados os possíveis ajustes que poderiam ser feitos para otimizar a experiência de usuários com diferentes necessidades. Isso inclui, por exemplo, a implementação de funcionalidades que permitam uma personalização maior da interface, como a possibilidade de mudar o esquema de cores ou aumentar o tamanho da fonte. Além do design e da usabilidade, outro aspecto crucial na avaliação foi a eficiência funcional da interface. A eficiência funcional se refere à capacidade da plataforma de permitir

que os usuários realizem suas tarefas de forma rápida e com o mínimo de esforço. Isso significa que a interface precisa ser direta e prática, sem exigir que os usuários percam tempo tentando encontrar a função desejada ou repetindo ações desnecessárias. Na avaliação da eficiência, foram considerados aspetos como o tempo que os usuários levam para completar uma tarefa e o número de interações necessárias para executar uma função específica. Embora o tempo de resposta de sistemas mais simples, como o da plataforma em questão, não seja uma grande preocupação, é importante observar como os usuários interagem com a interface em termos de velocidade e fluidez. Mesmo em uma aplicação relativamente simples, como a plataforma de gestão de produção, a eficiência da interface pode ser medida pela rapidez com que os usuários acessam as funcionalidades e pelas opções que têm para personalizar a interface de acordo com suas preferências. Outro fator relevante que foi avaliado é o feedback do usuário, que constitui uma das ferramentas mais importantes para refinar a experiência de uso da plataforma. O feedback pode ser obtido de várias maneiras, seja através de observação direta durante o uso da plataforma ou através de questionários aplicados aos usuários após a interação com o sistema. Nesse caso, o questionário foi utilizado como a principal forma de coleta de feedback, permitindo que os usuários compartilhassem suas percepções sobre a interface e sugerissem melhorias. Esse feedback é essencial para garantir que a interface atenda às expectativas dos usuários e que o sistema continue evoluindo com base nas necessidades reais de quem o utiliza. Concluindo, a avaliação da interface da Plataforma de Gestão e Previsão de Produção Alimentar visa assegurar que o design, a usabilidade e a eficiência da plataforma estejam otimizados para proporcionar uma experiência de uso agradável e produtiva. Ao seguir uma abordagem focada no usuário, a plataforma foi projetada para ser intuitiva e funcional, mas a avaliação contínua e o feedback dos usuários são indispensáveis para garantir que ela continue evoluindo de acordo com as necessidades dos seus utilizadores. A avaliação da interface, portanto, se baseia em critérios sólidos que consideram não apenas o que é funcional, mas também o que proporciona uma experiência eficiente e satisfatória.

Capítulo 4

Desenvolvimento

Neste capítulo, descreve-se a implementação do sistema de gestão de produção de leite, destacando os pontos relevantes, dificuldades enfrentadas e soluções técnicas aplicadas durante o desenvolvimento. Abordam-se detalhadamente como cada componente do sistema foi projetado para atender às necessidades específicas dos utilizadores, desde a interface intuitiva do frontend até a robustez do backend e a eficiência da base de dados. Além disso, discutem-se os desafios encontrados ao integrar diversas funcionalidades, como a criação e gestão de despesas e receitas, o monitoramento do stock de leite e a geração de relatórios detalhados. Exploram-se também as escolhas tecnológicas feitas para garantir a escalabilidade do sistema e sua adaptabilidade às exigências futuras, bem como as decisões de arquitetura que influenciaram positivamente a performance e a experiência do utilizador.

4.1 Arquitetura do Sistema

O sistema de gestão de produção de leite foi desenvolvido com uma arquitetura de três camadas, que promove modularidade, escalabilidade e facilidade de manutenção. A camadas frontend é responsável pela interface do usuário e interações. A camada de backend gere a lógica de negócios e o processamento de dados e base de Dados armazena e gere todos os dados do sistema.

Esta estrutura garante uma separação clara das responsabilidades, facilitando o desenvolvimento paralelo e a manutenção independente de cada camada. A Figura 4.1 mostra a tecnologia usada. Durante o desenvolvimento do sistema, surgiram vários desafios que

Componente	*Tecnologia*	*Versão*
Frontend	Flutter	2.5.0
Backend	Laravel	8.x
Base de Dados	MySQL	8.0
Linguagens	Dart (Frontend), PHP (Backend)	Dart 2.14.0, PHP 8.0
Controle de Versão	Git	2.33.0
CI/CD	Jenkins	2.303.2
Testes	PHPUnit (Backend), Flutter Test (Frontend)	N/A

Figura 4.1: Tecnologia Utilizada

exigiram soluções técnicas específicas para garantir a eficiência e a integridade do projeto. Um dos primeiros desafios foi a integração entre o Flutter e o Laravel, que apresentou dificuldades principalmente na comunicação segura entre o frontend e o backend. Para resolver essa questão, foi implementada uma autenticação baseada em tokens `JSON Web Token (JWT)`, o que garantiu que todas as requisições fossem devidamente autenticadas e seguras. Além disso, padronizou-se o formato das respostas da `API`, facilitando a interação entre os dois sistemas e garantindo uma comunicação consistente.

Outro desafio importante foi a modelagem de dados complexa, especialmente no que diz respeito à representação das relações entre os diferentes aspetos da produção de leite, como estoque, vendas e custos. Para resolver isso, foi utilizado um `schema` de base de dados normalizado, que permitiu organizar as informações de maneira eficiente e escalável. Além disso, foram aplicadas técnicas avançadas, como o uso de `views` e `stored procedures`, para realizar cálculos complexos diretamente no base de dados, melhorando a precisão e a performance dos processos de negócio.

A performance em dispositivos móveis também se mostrou um desafio, uma vez que o sistema precisava funcionar de maneira eficiente em aparelhos com diferentes capacidades

de processamento. Para garantir uma experiência de uso satisfatória, foram implementadas diversas otimizações no Flutter, incluindo o uso de técnicas de paginação na API, o que permitiu que grandes volumes de dados fossem carregados de forma incremental, reduzindo o tempo de resposta. Além disso, foi adotado o uso de caching tanto no frontend quanto no backend, o que minimizou a necessidade de realizar requisições repetidas ao servidor, melhorando significativamente a performance geral do sistema, mesmo em dispositivos com recursos limitados.

4.2 Desenvolvimento de Frontend

A Figura 4.2 apresenta a organização dos diretórios do frontend da plataforma móveis Milkpoint. A seguir, são descritos os principais arquivos e diretórios do frontend da aplicação. Este arquivo `main.dart` é a base de um aplicativo Flutter que lida com a

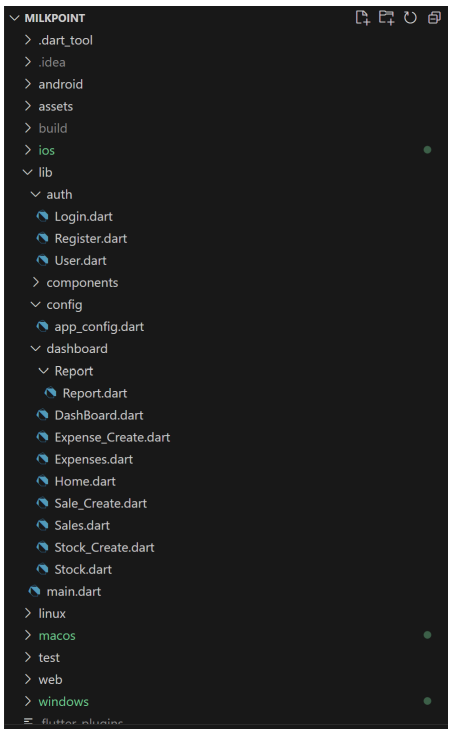


Figura 4.2: Estrutura de Diretorio de Frontend

autenticação de usuários e alterna entre diferentes telas com base no estado do usuário

logado. Ele usa o pacote `shared preferences` para armazenar e recuperar dados do usuário localmente e navega entre uma tela de `login` e um painel de controle (`Dashboard`), dependendo das informações recuperadas. O aplicativo começa com a função `main()`, que inicia o widget principal do Flutter, chamado `MyApp`. Esse widget é uma classe `StatelessWidget`, o que significa que sua estrutura não muda durante a execução. Dentro do `MyApp`, há um `MaterialApp`, que define a configuração geral do aplicativo, como o título e o tema, além de desabilitar a exibição do banner de depuração no modo de desenvolvimento. A tela inicial do aplicativo é definida pelo widget `MyHomePage`, que é um `` `StatefulWidget` ``. Um `StatefulWidget` permite que o estado mude dinamicamente durante o uso do aplicativo. O título da página é passado como um parâmetro para esse widget. O estado da `MyHomePage` é gerenciado pela classe `Start`, que é onde ocorre o carregamento dos dados do usuário. Quando o aplicativo inicia, o método `initState()` da classe `Start` é chamado, o que ativa o carregamento assíncrono do usuário salvo localmente com `SharedPreferences`. Este método usa a função `loadUser()`, que tenta recuperar os dados do usuário armazenados em JSON. Se esses dados existirem, eles são convertidos em um objeto da classe `` `User` ``, utilizando o método `User.fromJson()`. Se não houver dados, um usuário com nome padrão "a"é usado, o que indica que o login ainda não foi feito. O método `build()` determina qual tela será exibida com base no nome do usuário. Se o nome for "a"(indicando que os dados do usuário ainda não foram recuperados ou o login não foi feito), o aplicativo exibe a tela de login (`Login`). Caso contrário, ele exibe o painel de controle (`DashBoard`). Em resumo, o código alterna entre uma tela de login e uma tela de painel de controle com base nas informações de autenticação armazenadas no dispositivo, utilizando `shared preferences` para salvar e recuperar os dados do usuário de maneira persistente.

O arquivo `DashBoard.dart` define o funcionamento do painel de controle (`Dashboard`) de um aplicativo Flutter. Ele gerencia a navegação entre diferentes seções do aplicativo, como estoque, vendas e despesas, e permite funcionalidades como logout e exclusão de usuários por meio de uma API. O `Dashboard` utiliza o pacote `shared preferences`

para carregar e salvar informações do usuário localmente e `http` para fazer requisições à API. O arquivo começa com a importação de vários pacotes necessários, como `flutter/material.dart` para a criação da interface do usuário, pacotes locais para diferentes telas e funções do painel (como criação de vendas, relatórios, estoques e despesas), e `share dp references` para persistência de dados no dispositivo. O pacote `http` é utilizado para interagir com uma API remota, enquanto `app config.dart` contém configurações globais, como cores e a página selecionada. A classe principal do arquivo, `DashBoard`, é um `StatefulWidget`, o que significa que o estado da interface pode mudar durante o uso. Isso é necessário porque a interface exibe diferentes páginas e responde a ações do usuário, como a troca de abas e o logout. A lógica da interface e a gestão do estado são manipuladas pela classe `DashBoardState.` Dentro do estado, o código armazena informações importantes, como o usuário atual, carregado de forma assíncrona de `SharedPreferences` quando o painel é inicializado. Além disso, a interface é composta por uma lista de páginas (como `'Home' 'Stock'`, `'Sales'`, entre outras) que podem ser exibidas de acordo com a seleção feita na barra de navegação inferior. Quando o painel é iniciado, o método `initState()` é chamado. Ele configura a tela inicial e tenta carregar o usuário salvo localmente, o que permite a personalização do painel com base nas preferências e dados do usuário logado. Caso o usuário não esteja logado, a tela de login é exibida. A função `deleteUser()`é responsável por enviar uma requisição HTTP para eliminar um usuário da base de dados externa, usando uma API remota. Se o usuário for eliminado com sucesso, uma mensagem de confirmação é exibida. Em caso de erro, o código trata o problema e exibe uma mensagem de erro no console. Outra funcionalidade importante é o método `logout()`, que remove os dados do usuário de `SharedPreferences`, o que efetivamente desconecta o usuário e o redireciona para a tela de login. Isso é feito por meio de um diálogo de confirmação que é exibido ao usuário ao clicar no ícone de logout na barra superior. A interface gráfica é composta por um `Scaffold`, que organiza a estrutura básica da página, como a barra de navegação inferior e o corpo principal da tela. A barra de navegação permite que o usuário alterne entre as diferentes páginas do painel de controle. Cada ícone da barra representa uma área do aplicativo (como estoque

ou vendas), e a página correspondente é exibida ao ser selecionada. As cores e o estilo do painel são definidos pelo arquivo de configuração `app config.dart`, que centraliza a personalização da interface, como as cores de fundo e os ícones ativos. A navegação e a seleção de páginas são gerenciadas dinamicamente, permitindo que o conteúdo do painel mude conforme o usuário interage com a barra de navegação.

O arquivo `report.dart` contém a implementação de widgets para exibir relatórios de despesas e stock em diferentes intervalos de tempo (diário e mensal) utilizando Flutter. O objetivo principal deste arquivo é criar interfaces de utilizador para gerar e visualizar relatórios de despesas e stock de forma simples e direta. No código, o widget `HomeExpense` serve como a página inicial do sistema de relatórios. Ele apresenta quatro botões que permitem ao utilizador escolher entre diferentes tipos de relatórios:

- Relatório diário de despesas:direciona para a página onde o utilizador pode selecionar uma data específica para ver o relatório de despesas desse dia.
- Relatório mensal de despesas :leva o utilizador à página onde é possível escolher um mês e um ano para visualizar as despesas do mês selecionado.
- Relatório diário de Estoque :oferece a funcionalidade de selecionar uma data específica e ver os dados de Estoque desse dia.
- Relatório mensal de Estoque :semelhante ao anterior, mas permite selecionar o mês e o ano para gerar o relatório de estoque mensal

Na página de `DailyExpense`, o utilizador pode selecionar uma data através de um `DatePicker`. O valor da data escolhida é formatado e inserido num campo de texto. Quando o botão `See Report` é clicado, a função `See Report` é chamada, que constrói uma URL contendo a data selecionada e tenta abrir o link gerado. Caso o link não possa ser aberto, uma exceção é lançada. A página de `MonthlyExpense` utiliza dois menus dropdown para que o utilizador possa selecionar um mês e um ano. O dropdown de meses oferece uma lista de meses do ano, enquanto o dropdown de anos é gerado dinamicamente, incluindo 50 anos a partir de 2001. Após selecionar o mês e o ano, ao clicar em `'See Report'`, a

função `SeeReport` constrói a URL do relatório com os parâmetros adequados e tenta abrir a página. Da mesma forma, a página de `DailyStock` oferece uma interface para selecionar uma data e ver o relatório de estoque diário. A data é selecionada através de um `DatePicker,` e a função `SeeReport` constrói a URL com os valores selecionados e tenta abrir o relatório correspondente. Por fim, a página de **`MonthlyStock`** segue a mesma lógica das outras páginas mensais, permitindo ao utilizador selecionar um mês e ano para ver o relatório de estoque desse período. Os valores são capturados pelos menus dropdown, e a função *SeeReport* lança a Uniform Resource Locator (URL) gerada. Esses widgets utilizam uma interface visual consistente, com campos bem delineados, botões de ação e interatividade facilitada. As cores e estilos são ajustados conforme as preferências da aplicação (definidas por `AppConfig.prefs`). O objetivo é fornecer uma forma rápida e intuitiva de consultar os relatórios, acessando URLs geradas dinamicamente.

O arquivo `Home.dart`, que faz parte de uma aplicação Flutter. Este arquivo implementa a tela principal da aplicação, que inclui a exibição de dados analíticos e a capacidade de gerar relatórios. O código começa com a importação de bibliotecas necessárias, como `Material`, `http` e outras específicas da aplicação. Em seguida, a classe `Home` é definida como um *StatefulWidget*, gerenciando o estado da tela principal. Esta classe inicializa um objeto 'User' e busca dados de um API. Dentro da classe *HomeState*, várias variáveis são inicializadas e os dados do utilizador e do API são carregados. O método *getData* faz uma chamada HTTP para buscar informações analíticas, como receitas e despesas diárias e mensais. O método `loadUser` é responsável por recuperar as informações do utilizador armazenadas em `SharedPreferences`. A interface do utilizador é construída no método `build`, que inclui um `CircularProgressIndicator` para indicar que os dados estão a ser carregados, além de saudações personalizadas que mostram o nome do utilizador. Existem vários cartões (widgets) que exibem dados como `Daily IOFC`, `Monthly IOFC` e `Monthly Milk Production`.Um `FloatingActionButton` é adicionado à tela, permitindo ao utilizador abrir um modal para selecionar diferentes tipos de relatórios, como vendas, despesas e estoque. Para cada um desses relatórios, há um `GestureDetector` que, ao ser tocado, exibe um modal com opções para visualizar os dados diariamente ou

mensalmente. A estrutura de cada relatório é semelhante, com cartões de informação que mostram títulos e valores, estilizados com cores e bordas específicas. A função `buildCard` é utilizada para criar esses cartões, permitindo a reutilização de código e consistência na apresentação dos dados. o arquivo `Home.dart` implementa uma interface interativa onde o utilizador pode visualizar dados analíticos e selecionar relatórios, facilitando o acesso à informação relevante de maneira organizada e atraente.

O arquivo `Stock Create.dart` é um componente de uma aplicação Flutter que permite ao utilizador criar entradas para o stock de leite. Ele utiliza um `StatefulWidget` chamado `StockCreate`, que gere o estado da tela onde o utilizador pode inserir dados relacionados ao leite. No início do código, são importadas as bibliotecas necessárias, incluindo `Material`, `http` e componentes específicos da aplicação, como `app config` e `input`. Dentro da classe `StockCreateState`, são inicializados controladores de texto para os campos que vão armazenar a quantidade de litros de leite, a data e o número de vacas. Existem também variáveis booleanas que ajudam a controlar se os campos de entrada têm dados válidos ou não. Um método chamado `Create` é responsável por validar os dados inseridos. Para isso, existem funções internas que verificam se os campos de litros e vacas estão vazios, e se a data está no formato correto. Para validar a data, é utilizada uma expressão regular que garante que o formato está de acordo com o padrão `dd/mm/yyyy`. Quando o utilizador tenta criar uma nova entrada, o aplicativo exibe um `CircularProgressIndicator` para indicar que a operação está em progresso. Um pedido HTTP é enviado para a API correspondente para criar o estoque de leite, enviando os dados dos litros, da data e do número de vacas. Se o pedido for bem-sucedido (status code 200), o utilizador é redirecionado para o `DashBoard`. No método `initState`, a data atual é formatada e definida no controlador de texto da data. Há uma função que permite ao utilizador escolher uma data a partir de um `DatePicker`, que actualiza o campo de entrada da data conforme o utilizador faz a seleção. A interface do utilizador é construída no método `build`. Uma `SingleChildScrollView` permite que o conteúdo seja rolável, caso exceda a altura da tela. Dentro dela, existem elementos que incluem textos para título, campos de entrada para litros e número de vacas, além de um campo de entrada

para a data que é somente leitura e é preenchido ao tocar no campo. Por fim, há um botão `'CREATE'` que, quando pressionado, chama a função `Create` para submeter os dados. O botão está estilizado com cores e bordas definidas nas configurações da aplicação. o arquivo `Stock Create.dart` oferece uma interface intuitiva para que os utilizadores possam inserir e criar novas entradas de estoque de leite, validando as informações antes de enviá-las para uma API.

O arquivo `Stock.dart` é um componente de uma aplicação Flutter que exibe uma lista do stock de leite. A interface é construída como um `StatefulWidget` denominado `Stock`, que permite a manipulação do estado da aplicação. A classe de estado associada, chamada `StockState`, é onde a maior parte da lógica ocorre. No método `initState`, que é chamado quando o widget é criado, a função `getData` é invocada para buscar informações sobre o estoque de leite. Inicialmente, uma variável booleana chamada `isLoading` é configurada para `true` para indicar que os dados estão a ser carregados, e uma lista vazia chamada `datas` é inicializada para armazenar as informações obtidas. A função `getData` realiza uma chamada HTTP GET para a API, com o objetivo de obter a lista do estoque. Se a resposta for bem-sucedida, ou seja, se o código de status for igual a 200, os dados são processados. A resposta JSON é convertida e cada entrada é adicionada à lista `datas` Caso ocorra algum erro durante a chamada da API, uma mensagem de erro é impressa no console. Após a conclusão do carregamento dos dados, `isLoading` é definido como `false`, o que indica que o carregamento foi concluído. Outra função, chamada `deleteStock`, é responsável pela remoção de uma entrada específica do estoque. Esta função envia uma requisição HTTP POST para a API, com o ID do estoque que deve ser removido. Se a operação for bem-sucedida, a entrada correspondente é removida da lista `datas`. No método `build`, é configurado um *Scaffold* que proporciona a estrutura básica da interface. Um `CircularProgressIndicator` é exibido se os dados ainda estiverem a ser carregados. Caso contrário, a interface apresenta uma tabela que exibe os dados do estoque de leite. A tabela é criada utilizando um `DataTable`, que contém colunas para ID, litros, número de vacas e data. Há também uma coluna adicional que contém um ícone para deletar a entrada correspondente. Quando o utilizador clica neste ícone,

um `AlertDialog` aparece, solicitando a confirmação da ação de remoção. A interface do utilizador também inclui um `FloatingActionButton` que permite ao utilizador adicionar novas entradas ao estoque. Quando pressionado, este botão altera a variável `selectIndex` nas configurações da aplicação e redireciona o utilizador para o `DashBoard`.O arquivo `Stock.dart` fornece uma interface funcional para visualizar, adicionar e remover entradas de estoque de leite, com interações fluídas que garantem uma boa experiência para o utilizador.

O arquivo `Login.dart` define um componente de interface em Flutter que permite aos utilizadores autenticar-se na aplicação. Este componente é um `StatefulWidget` chamado `Login`, e a sua lógica de estado é gerida na classe `LoginState`. Dentro da classe de estado, são definidas variáveis que controlam a interação do utilizador. Dois controladores de texto são criados, um para o email e outro para a palavra-passe, ambos usados para capturar a entrada do utilizador. Também existe uma variável booleana chamada `error`, que indica se ocorreu um erro durante o processo de login. A função `saveUser` é responsável por armazenar os detalhes do utilizador, utilizando a biblioteca `SharedPreferences`. Quando o login é bem-sucedido, os dados do utilizador são convertidos em formato JSON e salvos localmente, permitindo que a aplicação memorize as informações do utilizador para futuras sessões. A função `Enter` é a responsável por realizar a autenticação do utilizador. Quando chamada, ela exibe um `CircularProgressIndicator` no ecrã para indicar que o carregamento está a ocorrer. Em seguida, uma requisição HTTP POST é enviada para a API, com o email e a palavra-passe do utilizador. Se a resposta for bem-sucedida, ou seja, se o código de status for igual a duzentos, a aplicação verifica se o nome do utilizador foi retornado. Se sim, os dados do utilizador são utilizados para criar um objeto `User`, que é então salvo através da função `saveUser`. Após isso, o utilizador é redirecionado para o `DashBoard`. Caso a autenticação falhe, e a resposta seja um código de status quatrocentos ou quatrocentos e um, a variável `error` é definida como `true`, fazendo com que uma mensagem de erro seja apresentada no campo da palavra-passe. Se a resposta retornar qualquer outro código de erro, a variável `error` é definida como `false`, e uma mensagem de erro é impressa no console. No método `build`, é criada a estrutura visual da interface.

Um `Scaffold` é utilizado para fornecer a base, e a cor de fundo é definida com base nas preferências da aplicação. O cabeçalho da aplicação é configurado para não exibir um botão de retroceder. Dentro do corpo da aplicação, um `SingleChildScrollView` permite que o conteúdo seja rolável, especialmente em dispositivos com ecrãs pequenos. O layout inclui uma imagem que representa o logotipo da aplicação, seguida por campos de entrada para o email e a palavra-passe. O campo da palavra-passe exibe uma mensagem de erro se a autenticação falhar. Um botão para realizar o login é apresentado, e quando pressionado, chama a função `Enter`. A interface também inclui uma mensagem que pergunta ao utilizador se já possui uma conta, juntamente com um link que redireciona para a página de registo caso o utilizador ainda não tenha uma conta. O arquivo `Login.dart` proporciona uma experiência de autenticação simples e intuitiva, permitindo que os utilizadores façam login na aplicação de forma eficiente, enquanto também oferece a opção de registo para novos utilizadores.

O arquivo `register.dart` é responsável por implementar a funcionalidade de registo de utilizadores na aplicação. Ele utiliza o framework Flutter e interage com a biblioteca Firebase para gerir a autenticação. No início do ficheiro, são importadas as bibliotecas necessárias, incluindo aquelas que fornecem suporte à autenticação e à gestão de estados. A classe principal que contém a lógica do registo é um `StatefulWidget`, o que permite que o estado da interface do utilizador seja mantido e atualizado dinamicamente. Dentro desta classe, existe um formulário que permite ao utilizador introduzir informações, como o endereço de email e a palavra-passe. O formulário inclui validações para garantir que os dados inseridos são válidos. Por exemplo, verifica se o endereço de email está no formato correto e se a palavra-passe cumpre requisitos mínimos de segurança. Quando o utilizador clica no botão de registo, é chamada uma função que tenta criar uma nova conta no Firebase utilizando os dados fornecidos. Se o registo for bem-sucedido, o utilizador é redirecionado para a página principal da aplicação. Caso contrário, são apresentadas mensagens de erro que informam o utilizador sobre o que correu mal, como, por exemplo, se o e-email já está associado a outra conta ou se ocorreu um erro de rede. O arquivo também inclui uma opção para que o utilizador possa voltar à página de login, caso já

tenha uma conta. Essa funcionalidade proporciona uma melhor experiência ao utilizador, permitindo uma navegação intuitiva na aplicação.

4.3 Desenvolvimento de Backend

No projeto MilkPoint API, o backend ou API é uma parte crucial da arquitetura cliente-servidor, encarregue de processar os dados e aplicar as regras de negócio da aplicação. Quando uma requisição é recebida, o backend comunica-se com a base de dados para processar a informação necessária e responder de forma adequada às solicitações dos utilizadores. Esta separação entre a aplicação do lado do cliente e o serviço de backend contribui para um desempenho mais eficiente, especialmente em situações de elevada carga de processamento. Tal como no caso de muitas aplicações modernas, o MilkPoint API utiliza a arquitetura de API REST para facilitar a comunicação entre o frontend e o backend. Através de endpoints acessíveis via HTTP, o backend permite a execução de operações como criação, leitura, atualização e eliminação de dados CRUD, de forma estruturada e escalável. Esta abordagem aumenta a modularidade e facilita a manutenção do sistema ao longo do tempo. Para o desenvolvimento do backend foi utilizado o framework Laravel, um dos mais populares no ecossistema PHP. Laravel é conhecido pela sua simplicidade e eficiência, permitindo o desenvolvimento de API REST de forma rápida e organizada. O Laravel também fornece uma série de ferramentas integradas, como o Eloquent ORM, que facilita a interação com a base de dados ao permitir que as operações sejam realizadas de forma mais intuitiva, sem necessidade de escrever consultas SQL manuais.

4.3.1 Estrutura de diretório

A estrutura de diretórios de um projeto é fundamental para a organização e manutenção do código a longo prazo. No MilkPoint API, seguimos as convenções recomendadas pelo Laravel, o que ajuda a manter uma separação clara entre as diferentes responsabilidades da aplicação.

Na imagem apresentada 4.3, pode-se ver a estrutura do diretório do backend.

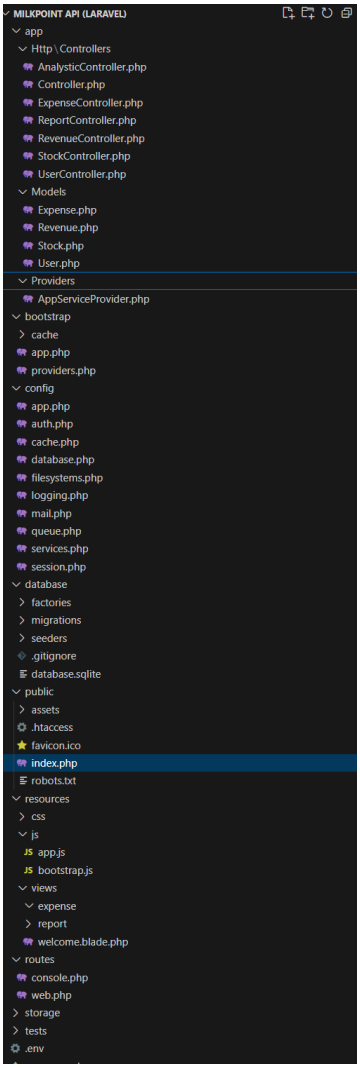


Figura 4.3: Estrutura de diretório do projeto do backend.

O arquivo `index.php` é o ponto de entrada principal da aplicação Laravel e é responsável por iniciar o processo de tratamento das requisições recebidas. Quando um utilizador acede à aplicação, este ficheiro é executado primeiro. Ele começa por verificar se a aplicação está em modo de manutenção, carregando o ficheiro de manutenção, caso exista. Em seguida, regista o autoloader do Composer, que carrega automaticamente todas as dependências da aplicação. Por fim, o ficheiro inicializa a aplicação Laravel ao invocar o ficheiro `bootstrap/app.php` e captura a requisição HTTP recebida, passando-a para o núcleo do Laravel, que a processa e devolve a resposta adequada ao utilizador.

Já o dirétorio `app/Http/Controller` contém as classes responsáveis por gerir as requisições e respostas da aplicação Laravel. Estas classes, conhecidas como controladores, desempenham um papel essencial ao receber as solicitações `HTTP`, validar os dados fornecidos, configurar os métodos adequados para cada operação e definir os endpoints da API. Os controladores também são responsáveis por definir rotas que podem incluir autenticação e autorização, garantindo que apenas utilizadores autenticados possam aceder a determinados recursos. Além disso, os controladores encaminham as solicitações para os serviços ou modelos apropriados, que executam a lógica de negócio necessária para resolver a requisição, e retornam uma resposta adequada ao cliente ou utilizador. Assim, os controladores funcionam como o ponto central de ligação entre as requisições do frontend e a lógica de backend, assegurando uma comunicação eficiente e a execução correta das operações dentro do sistema.

Já o diretório `app/Models` armazena as classes que representam os modelos da aplicação, responsáveis pela interação com a base de dados através do ORM Eloquent. Os modelos são uma representação das entidades da base de dados, como por exemplo utilizadores, produtos ou pedidos, permitindo que o Laravel trate os dados como objetos em vez de tabelas ou colunas. Estes modelos são usados para realizar operações como consultas, inserções, atualizações e eliminações na base de dados de forma eficiente e intuitiva, sem a necessidade de escrever SQL manual. Através dos modelos, é possível definir relações entre diferentes entidades da base de dados como relações um-para-um, um-para-muitos ou muitos-para-muitos e também aplicar regras de validação e lógica de negócio associada

a cada entidade, tornando o desenvolvimento mais organizado e fácil de manter ao longo do tempo.

Já o diretório `config/` abriga todos os ficheiros de configuração, incluindo a configuração do sistema de autenticação, base de dados, mail, e outros serviços essenciais. O diretório `database/` é o diretório que contém migrações, seeders e outras ferramentas para gerir o esquema da base de dados. Finalmente, o diretório `routes/` é o local onde as rotas da API são definidas, especificando como as URL da aplicação mapeiam para os controladores.

4.3.2 Controladores

Os controladores são componentes fundamentais na arquitetura de software, especialmente em aplicações web que seguem o padrão Model-View-Controller (MVC). Eles desempenham um papel crucial ao gerir a lógica de interação entre o frontend e a lógica de negócios da aplicação. Os controladores são responsáveis por receber requisições HTTP, processá-las e retornar as respostas adequadas aos clientes. Quando uma requisição chega ao servidor, o controlador é o responsável por extrair os dados relevantes, como parâmetros de URL, cabeçalhos e corpo da requisição. A validação dos dados recebidos é uma parte essencial do trabalho do controlador, garantindo que as informações estejam corretas e no formato esperado, o que ajuda a prevenir erros e a garantir a integridade dos dados. Após a validação, o controlador invoca serviços ou modelos que contêm a lógica de negócios necessária para processar a requisição. Isso permite que os controladores permaneçam leves e focados no fluxo de dados, evitando que a lógica de negócios se acumule dentro deles. Quando o processamento está completo, o controlador prepara a resposta adequada, que pode ser em formatos como JSON, dependendo do tipo de requisição. As respostas podem incluir dados solicitados, mensagens de erro ou códigos de status HTTP, conforme apropriado.

Além disso, os controladores implementam lógica para tratar erros, como situações em que os dados são inválidos ou um recurso não é encontrado. Isso envolve o retorno

de mensagens de erro claras e códigos de status HTTP que informam o cliente sobre o resultado da operação.

Os controladores são tipicamente organizados em classes, onde cada classe representa um conjunto específico de operações relacionadas a um recurso, como na imagem 4.4 `StockController` para operações sobre estoques. Os métodos dentro dessas classes correspondem a diferentes operações (endpoints) que podem ser realizadas sobre o recurso, utilizando métodos HTTP como `GET, POST, PUT, PATCH e DELETE.`. Estão na tabela 4.2. Um controlador em Laravel é uma classe que gerencia as requisições HTTP e a lógica de aplicação associada. Ele atua como intermediário entre os modelos (que representam os dados) e as views (que apresentam esses dados ao usuário). Os controladores normalmente estão localizados no `namespace AppHttpControllers`. Cada `método` dentro do controlador corresponde a uma ação específica, como `listar, criar, atualizar ou excluir` um recurso. Essa estrutura permite que o controlador organize e processe as requisições de forma eficiente, facilitando a interação entre a lógica de negócios e a apresentação dos dados. A tabela 4.1 apresenta as utilidades de cada método do controlado `StockController`.

Método	**Ação**	**Descrição**
index()	Listar Estoques	Retorna todos os registos de estoque em formato JSON. Ou seja é um *GET*
store()	Criar Estoque	Recebe dados e cria um novo registo de estoque. Ou seja é um *POST*
show($ id)	Mostrar Estoque Específico	Retorna um registo de estoque específico pelo ID.. Ou seja responda um *GET* com id
update()	Atualizar Estoque	Usado para atualizar os dados de um estoque existente. Ou seja responda um *PUT* ou *PATCH* com o ID e os novos dados.
destroy($ id)	Deletar Estoque	Utilizado para eliminar um estoque. Responde a uma requisição *DELETE* com o ID do estoque na URL.

Tabela 4.1: Métodos http da biblioteca MVC do Laravel do Controlador de Estoque.

Um `endpoint` na tabela 4.2 é um ponto de acesso em um serviço web que permite

```php
<?php

namespace App\Http\Controllers;

use App\Http\Controllers\Controller;
use App\Models\Stock;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Validator;

class StockController extends Controller
{
    public function list(Request $request)
    {
        // Listar Produção em ordem descendente
        $stocks = Stock::orderBy('id', 'desc')->get();
        return response()->json($stocks);
    }

    public function create(Request $request)
    {
        // Validar os dados necessários para criar as Produções
        $validator = Validator::make($request->all(), [
            'liter' => 'required|string|max:255',
            'date' => 'required|string|max:255',
            'cows' => 'required|string|max:255',
        ], [
            'liter.required' => 'The liter field is required.',
            'date.required' => 'The date field is required.',
            'cows.required' => 'The cows field is required.',
        ]);

        if ($validator->fails()) {
            return response()->json([
                'error' => true,
                'message' => $validator->errors()->first()
            ], 400);
        }

        // Criar Produções
        $stock = Stock::create([
            'liter' => $request->liter,
            'date' => $request->date,
            'cows' => $request->cows,
        ]);
        // Retornar uma Produção recem criada em JSON
        return response()->json($stock);
    }

    public function delete($id)
    {
        // Verificar a existencia da Produção
        $stock = Stock::find($id);

        if (!$stock) {
            return response()->json([
                'error' => true,
                'message' => 'Stock does not exist'
            ], 404);
        }
        // Eliminar a Produção
        $stock->delete();

        return response()->json([
            'success' => true,
            'message' => 'Stock deleted successfully'
        ]);
    }
```

Figura 4.4: Controlador de Estoque.

a comunicação entre um cliente e um servidor. Em termos práticos, é uma URL específica onde um recurso ou serviço pode ser acessado por meio de métodos HTTP (como GET, POST, PUT, DELETE). A API desenvolvida para a aplicação possui uma série de endpoints que permitem a interação com diversos recursos, facilitando operações como autenticação, gestão de utilizadores, controlo de stock, receitas, despesas e geração de relatórios.

Funcionalidade	**Método**	**Endpoint**	**Descrição**
Autenticação	POST	*/user/authenticate*	Autentica um utilizador e retorna um token de acesso
Gestão de Utilizadores	GET	*/user/list*	Lista todos os utilizadores
-	POST	*/user/create*	Cria um novo utilizador
-	POST	*/user/delete/id*	Remove um utilizador pelo ID
Gestão de Stock	GET	*/stock/list*	Lista todos os itens em stock
-	POST	*/stock/create*	Cria um novo item de stock
-	POST	*/stock/delete/id*	Remove um item de stock pelo ID
Gestão de Receitas	GET	*/revenue/list*	Lista todas as receitas
-	POST	*/revenue/create*	Cria uma nova receita
-	POST	*/revenue/delete/id*	Remove uma receita pelo ID
Gestão de Despesas	GET	*/expense/list*	Lista todas as despesas
-	POST	*/expense/create*	Cria uma nova despesa
-	POST	*/expense/delete/id*	Remove uma despesa pelo ID
Relatórios	GET	*/report/revenue*	Gera relatório de receitas
-	GET	*/report/expense*	Gera relatório de despesas
-	GET	*/report/stock*	Gera relatório de stock
-	GET	*/report/iofc*	Gera relatório de receita líquida

Tabela 4.2: Endpoints. .

4.4 Previsão

A implementação do sistema de previsão do IOFC foi desenvolvida com o framework Laravel em PHP, focando na criação de uma API RESTful para processar e gerar previsões. O núcleo da implementação é o controlador `ForecastController.php`, que está localizado no diretório na figura 4.5 `app/Http/Controllers/`. Esse controlador é responsável por gerenciar as requisições de previsão e por implementar a lógica de cálculo, seguindo o padrão de arquitetura MVC e aplicando boas práticas de programação. O algoritmo de

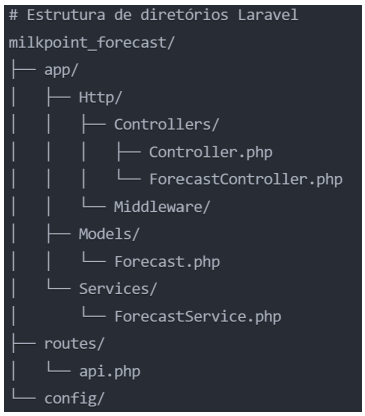


Figura 4.5: Diretório de Previsão

previsão implementado utiliza a técnica de média móvel para gerar previsões, operando em duas etapas principais. Na primeira etapa, um loop externo `($i)` define o período futuro a ser previsto, enquanto um loop interno `($j)` calcula a média móvel dos dados históricos. Esse algoritmo recebe um conjunto de dados históricos de IOFC, utilizando uma janela móvel de tamanho fixo para processar os últimos sessenta dias e gerar as previsões. No processamento, ele acumula as somas e contagens dos dados válidos, desconsiderando valores nulos para evitar distorções nas previsões. A partir dessas médias calculadas, o sistema gera as previsões dos próximos quinze dias, utilizando os padrões históricos observados para projetar tendências futuras de forma suavizada.

Aspectos técnicos importantes incluem a configuração de parâmetros essenciais ao modelo. `windowSize` O tamanho da janela de análise define quantos dias de dados históricos são considerados no cálculo da média móvel. O parâmetro `dataCount` define a quantidade

de registros históricos disponíveis, enquanto `forecastDays` representa o número de dias para os quais a previsão é gerada, que neste caso é de quinze dias. Além disso, o sistema conta com validação de dados de entrada, remoção de valores nulos e normalização de valores extremos para garantir a consistência das previsões.

Para otimizar a performance, a implementação inclui o uso de cache para armazenar dados acessados com frequência, a validação dos parâmetros de entrada para evitar erros e o tratamento de exceções, aumentando a robustez do sistema. Além disso, foram aplicadas melhorias no desempenho, como a otimização dos loops de processamento e a redução de operações redundantes. Durante a implementação, alguns desafios foram encontrados, como erros de acesso a índices inexistentes e a precisão das previsões. Esses problemas foram solucionados com a implementação de verificações de limites de array e com ajustes nos parâmetros e nas médias móveis ponderadas para melhorar a acurácia das previsões. A implementação atual demonstra ser eficaz na geração de previsões consistentes, no processamento eficiente de dados históricos e na visualização clara dos resultados. No entanto, ela apresenta algumas limitações, como a sensibilidade a valores extremos, a necessidade de dados históricos consistentes e o ajuste manual de certos parâmetros. Para o futuro, é recomendada a adoção de algoritmos mais avançados, a automação no ajuste de parâmetros, melhorias na interface de visualização e a expansão das funcionalidades de análise, a fim de aumentar a precisão e flexibilidade do sistema.

O sistema expõe seus endpoints através de uma API REST , conforme a Figura mostrada C.1 o `historyDays`: número de dias históricos (padrão: 60) e o `forecastDays`: número de dias para previsão (padrão: 15)

Os resultados conforme a Figura C.2 são visualizados através de um gráfico que apresenta tanto os dados históricos, representados pela linha turquesa, quanto as previsões, ilustradas pela linha rosa. A linha turquesa representa os valores reais do IOFC ao longo dos sessenta dias anteriores, mostrando as variações naturais do mercado, enquanto a linha rosa projeta os valores futuros, apresentando uma tendência suavizada e baseada no histórico.

Capítulo 5

Testes e Discussão

Neste capítulo serão discutidos a avaliação e os resultados obtidos sobre a proposta de plataforma para gestão e previsão de produção alimentar. Na Secção 5.1 é apresentado o cenário no qual foi realizada a avaliação, na Secção 5.2 os resultados obtidos a partir da análise quantitativa e qualitativa dos questionários, e, por fim, na Secção 5.3 as considerações finais referentes a este capítulo.

5.1 Cenário de avaliação

Para a avaliação da plataforma, foram selecionados alunos de várias instituições de ensino, incluindo o , a Universidade de Trás-os-Montes e Alto Douro (UTAD), a Universidade de Évora e o Institute of Business (IOB), abrangendo diferentes níveis académicos. Ao todo, participaram 12 alunos, incluindo estudantes de Doutoramento em Informática, Mestrado em Informática e Mestrado em Engenharia Eletrotécnica e de Computadores.

A avaliação consistiu na instalação e exploração da aplicação, cujo processo foi simplificado para que os participantes realizassem tarefas como o registo de utilizador, a adição de stock, registo de novas vendas e despesas, bem como a visualização e download de relatórios de IOFC e receitas. Como a aplicação não requer uma interação prolongada ou complexa, o tempo estimado para explorar todas estas funcionalidades variou entre 5 a 10 minutos.

Os testes foram realizados em smartphones Android.

5.2 Análise e resultados dos questionários

A Figura mostra 5.1 O perfil dos participantes revelou que 91,7% tinham experiência prévia com desenvolvimento de software e plataformas móveis, indicando uma amostra qualificada para avaliar os aspetos técnicos da plataforma. A autoavaliação do conhecimento técnico mostrou uma distribuição equilibrada: 33,3% dos participantes classificaram o seu conhecimento como "Ótimo", 33,3% como "Bom"e 33,3% como "Neutro.

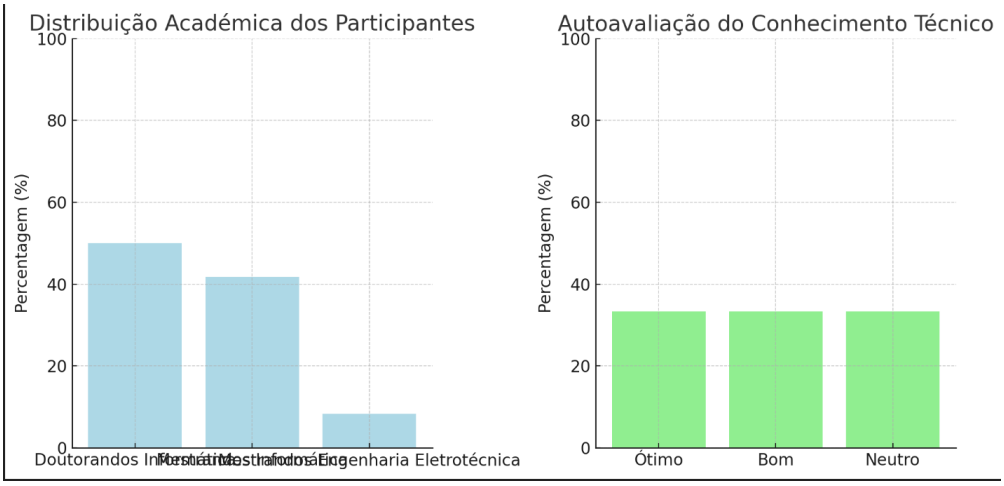


Figura 5.1: Perfil Académico e Técnico dos Participantes na Avaliação da Plataforma

O gráfico apresenta 5.2 os Índices de Satisfação através de barras em tom lilás. A escala vertical vai de zero até cinco pontos. A Interface apresenta uma pontuação entre quatro e cinco, mostrando-se bastante satisfatória. A Usabilidade revela valores igualmente positivos, também entre quatro e cinco pontos. A Performance destaca-se como o aspecto mais bem avaliado, aproximando-se mais do cinco. A Eficácia demonstra resultados muito bons, situando-se também entre quatro e cinco pontos. Por fim, o índice Geral mantém-se consistente com os demais indicadores, revelando uma avaliação igualmente positiva. De modo geral, todos os aspectos avaliados apresentam resultados muito satisfatórios, com pequenas variações entre si. A Performance sobressai ligeiramente como o ponto mais forte, enquanto os restantes indicadores mantêm-se em níveis muito próximos,

demonstrando uma consistência notável na avaliação global.

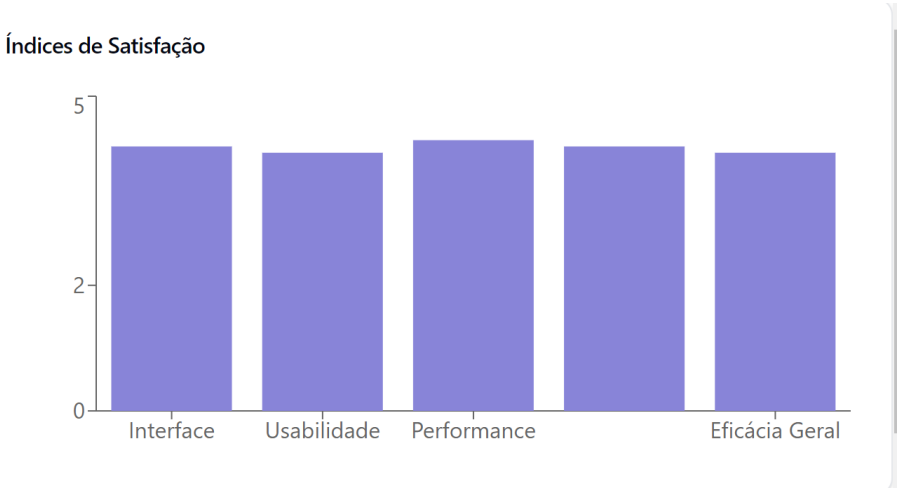


Figura 5.2: Indíce de Satisfação

Conforme apresentado na análise da Facilidade de Instalação 5.3, os resultados demonstram um cenário extremamente positivo relativamente ao processo de implementação. Os dados revelam que mais de dois terços dos inquiridos classificaram a instalação como "Muito Fácil"ou "Fácil", distribuídos equitativamente entre estas duas categorias. Uma parcela menor dos participantes manteve uma posição neutra face ao processo, enquanto é especialmente relevante notar a ausência total de avaliações negativas na categoria "Muito Difícil". Esta distribuição evidencia o sucesso significativo da estratégia de desenvolvimento voltada para a experiência do utilizador, demonstrando que o processo de instalação foi adequadamente planeado e executado, resultando numa experiência positiva para a maioria dos utilizadores. Como ilustrado no gráfico circular, a simetria entre as avaliações positivas, combinada com a ausência de feedback negativo, sugere uma robustez considerável no desenho da solução, confirmando a eficácia das metodologias implementadas no processo de instalação.

A presente análise da figura 5.4 reflete os resultados da recolha de feedback qualitativo dos utilizadores da Plataforma de Gestão de Produção Alimentar. O processo de avaliação

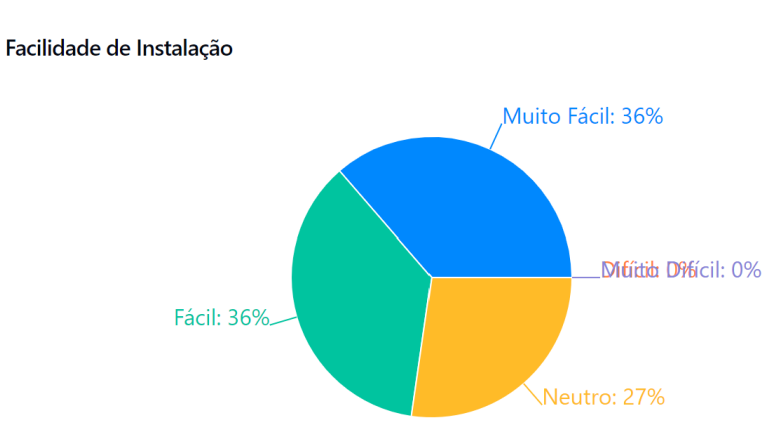


Figura 5.3: Facilidade de Instalação

identificou tanto pontos fortes como oportunidades de melhoria significativas, proporcionando uma base sólida para futuras atualizações do sistema. A avaliação do feedback demonstrou que a plataforma possui bases sólidas, evidenciadas pela satisfação dos utilizadores com determinados aspetos fundamentais, nomeadamente o tempo de resposta rápido e a facilidade na gestão de dados. « No entanto, na figura 5.5 foram identificadas

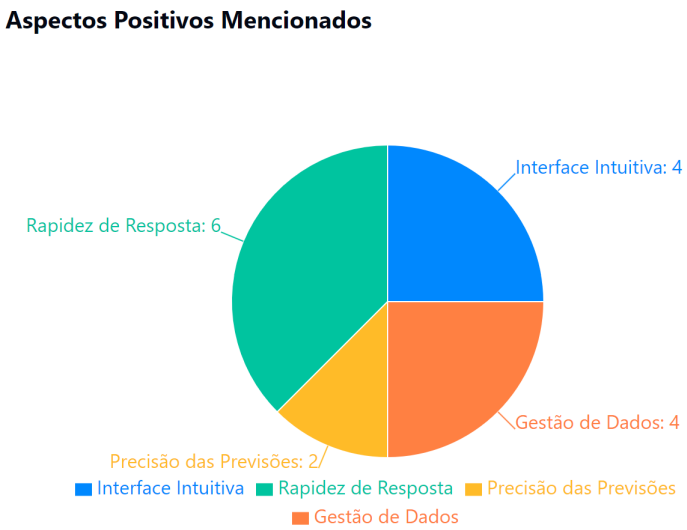


Figura 5.4: Aspecto Positivo

áreas específicas que requerem atenção e desenvolvimento. A questão mais significativa,

mencionada por oito utilizadores, prende-se com a compatibilidade de dispositivos. A limitação atual de funcionamento exclusivo em ambiente Android representa uma barreira significativa à adoção mais ampla da plataforma. Esta restrição foi consistentemente apontada como um obstáculo prioritário a ser ultrapassado. No âmbito da experiência do utilizador, a interface atual recebeu avaliações mistas. Enquanto alguns utilizadores destacam a sua intuitividade, outros identificaram pontos específicos para melhoria, como a visibilidade da palavra-passe durante o processo de registo. Esta divergência de opiniões sugere a necessidade de um refinamento focado em áreas específicas, mantendo os elementos que já funcionam eficazmente.A funcionalidade de previsões emerge como um recurso valorizado pelos utilizadores, embora existam sugestões para melhorar a sua precisão. Três utilizadores especificamente mencionaram este aspeto, indicando uma área de desenvolvimento potencialmente impactante para futuras atualizações.

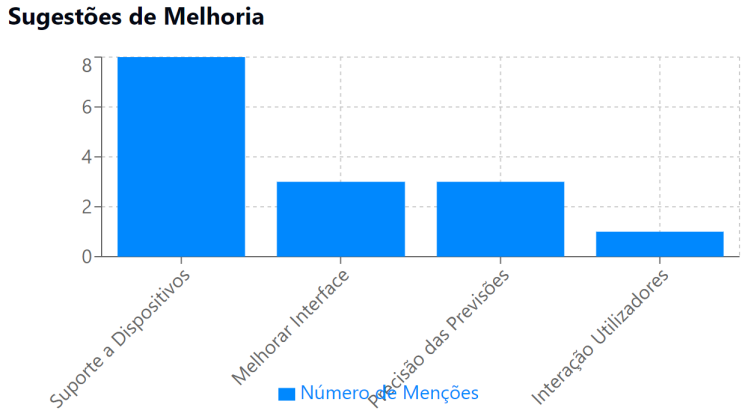


Figura 5.5: Sugestões de Melhoria

Com base na análise realizada, recomenda-se uma abordagem faseada para as melhorias: A prioridade máxima deve ser atribuída ao desenvolvimento de uma solução multiplataforma que elimine as atuais restrições de dispositivos. Em paralelo, sugere-se a implementação de melhorias pontuais na interface do utilizador, começando pela funcionalidade de visualização da palavra-passe durante o registo. Num segundo momento, deve-se focar no refinamento dos algoritmos de previsão e na expansão das funcionalidades

de interação entre utilizadores. Estas melhorias, embora menos urgentes, contribuirão significativamente para uma experiência mais completa e satisfatória. Por fim, Para garantir a evolução sustentada da plataforma, recomenda-se a implementação de um processo de integração mais abrangente e o estabelecimento de canais permanentes de feedback dos utilizadores. Esta abordagem permitirá uma evolução contínua alinhada com as necessidades reais dos utilizadores.

5.3 Discussão

A avaliação da plataforma para gestão e previsão de produção alimentar apresentou resultados significativamente positivos, com insights importantes obtidos através de uma amostra diversificada de 12 participantes de diferentes instituições de ensino superior. Os resultados quantitativos demonstraram um alto nível de satisfação dos usuários em diversos aspetos. A performance destacou-se como o ponto mais forte da plataforma, seguida por avaliações igualmente positivas nos quesitos de interface, usabilidade e eficácia. É relevante notar que 91,7% dos participantes possuíam experiência prévia com desenvolvimento de software e plataformas móveis, o que confere maior credibilidade à avaliação técnica realizada. Em relação à facilidade de instalação, a plataforma obteve uma avaliação notavelmente positiva, com mais de dois terços dos participantes classificando o processo como "Muito Fácil"ou "Fácil". A ausência total de avaliações na categoria "Muito Difícil"reforça o sucesso da estratégia de desenvolvimento focada na experiência do utilizador. No entanto, a avaliação também identificou oportunidades importantes de melhoria. A principal limitação apontada foi a compatibilidade exclusiva com dispositivos Android, sendo mencionada por oito participantes como uma barreira significativa para a adoção mais ampla da plataforma. Adicionalmente, foram sugeridas melhorias na interface do utilizador e no refinamento do sistema de previsões. Como perspetivas futuras, recomenda-se priorizar o desenvolvimento de uma solução multiplataforma, seguido por

melhorias pontuais na interface e no refinamento dos algoritmos de previsão. A implementação de um processo de integração mais abrangente e o estabelecimento de canais permanentes de feedback dos utilizadores também se mostram essenciais para garantir a evolução sustentada da plataforma. Em suma, a avaliação demonstrou que a plataforma possui uma base sólida e cumpre seus objetivos principais, mas apresenta oportunidades claras de evolução para melhor atender às necessidades dos utilizadores.

Capítulo 6

Conclusão

Este trabalho centrou-se no desenvolvimento de uma plataforma de Gestão e Previsão de Produção Alimentar, com especial enfoque na produção de laticínios. O sistema criado permite que os produtores organizem e controlem de forma mais eficiente as suas operações, como a gestão de stocks, o registo de receitas e despesas, bem como o cálculo do indicador económico IOFC. As funcionalidades foram desenvolvidas para serem acessíveis através de uma aplicação móvel, facilitando assim a gestão diária das atividades. Além disso, foram implementados mecanismos de previsão financeira baseados em dados históricos, proporcionando aos produtores ferramentas para uma análise mais estratégica dos custos e receitas.

Embora tenha sido dado um passo significativo com a estruturação da plataforma, a integração de métodos avançados de previsão ou IA não foi ainda implementada nesta fase. O foco centrou-se na criação da base da plataforma, deixando a aplicação de IA como uma expansão futura que poderá melhorar substancialmente as previsões de rendimento.

No futuro, o trabalho poderá ser ampliado com a reorganização das funcionalidades da plataforma, de forma a otimizar ainda mais o fluxo de trabalho e a experiência dos utilizadores. Também será importante expandir a aplicação para outros dispositivos, como o iPhone Operating System (IOS), garantindo uma maior abrangência de utilizadores. Adicionalmente, a integração de algoritmos de IA será um passo crucial para aumentar

a precisão das previsões, utilizando modelos mais avançados de aprendizagem automática. Com estas melhorias, a plataforma estará mais bem equipada para apoiar de forma completa a gestão e previsão da produção alimentar, respondendo às necessidades dos produtores e facilitando uma tomada de decisão mais eficaz e informada.

Bibliografia

[1] S. Patra, "A Review on Impact of Mobile Apps in Agriculture," Dezembro de 2023, Accessed: 2024-07-04.

[2] I. Kutyauripo, M. Rushambwa e L. Chiwazi, "Artificial intelligence applications in the agrifood sectors," 2023, Accessed: 2024-07-04.

[3] FAO, *FAO in Portugal*, https://www.fao.org/portugal/acerca-de/pt/, Accessed: 2024-07-01, jul. de 2024.

[4] A. Brum, M. Fagundes, J. Sausen, M. Casali e M. M. Brizolla, "As práticas sustentáveis na cadeia produtiva do leite e os objetivos de desenvolvimento sustentável," *Revista Campo-Território*, vol. 16, pp. 31–65, abr. de 2021. DOI: 10.14393/RCT164002.

[5] L. Oliveira, M. Pereira, N. Apolidório, S. Luciano, G. Possenti e R. Battisti, "AVALIAÇÃO DA PRODUÇÃO DE BIOGÁS A PARTIR DA DIGESTÃO ANAERÓBIA DE RESÍDUOS ORGÂNICOS ALIMENTARES ORIUNDOS DA CANTINA DO IFSC CÂMPUS CRICIÚMA," nov. de 2023.

[6] D. Documentation, *DelPro*, https://www.delaval.com/pt-br/nossas-solucoes/gerenciamento-de-rebanho/delaval-delpro/applications/delpro-farmmanager/, Accessed: 2024-07-02.

[7] M. Santos, R. Faoro, J. Matte et al., "Mapeamento de processos de uma empresa de laticínios," *Saber Humano Revista Científica da Faculdade Antonio Meneghetti*, vol. 9, pp. 84–104, dez. de 2019. DOI: 10.18815/sh.2019v9n15.360.

[8] AgroCR, *Os desafios enfrentados pelos agricultores no uso de tecnologia no campo!* https://pt.linkedin.com/pulse/os-desafios-enfrentados-pelos-agricultores-uso-de-tecnologia-campo-, Accessed: 2024-07-02, Fevereiro de 2023.

[9] S. Patra, "A Review on Impact of Mobile Apps in Agriculture," vol. 15, pp. 85–91, dez. de 2023.

[10] M. E. Marwa, J. Mburu, R. E. J. Oburu, O. A. Mwai e S. Kahumbu, "Impact of ICT Based Extension Services on Dairy Production and Household Welfare: The Case of iCow Service in Kenya," *The Journal of Agricultural Science*, vol. 12, p. 141, 2020. URL: https://api.semanticscholar.org/CorpusID:211130348.

[11] R. K. K. Sathya, *Time Series Analysis on Agricultural Commodity Prices*, https://www.researchgate.net/publication/351582113_Time_Series_Analysis_on_Agricultural_Commodity_Prices/fulltext/609e7fb8a6fdcccacb54e9d3/Time-Series-Analysis-on-Agricultural-Commodity-Prices.pdf?origin=scientificContributions, Accessed: 2024-07-04, Maio de 2021.

[12] F. O. Usman, "AI-DRIVEN PREDICTIVE ANALYTICS IN AGRICULTURAL SUPPLY CHAINS: A REVIEW: ASSESSING THE BENEFITS AND CHALLENGES OF AI IN FORECASTING DEMAND AND OPTIMIZING SUPPLY IN AGRICULTURE," Fevereiro de 2024, Accessed: 2024-07-04.

[13] F. Media, *A sustainable and food secure world for all*, https://https://www.fao.org/home/en, Accessed: 2024-07-02, jul. de 2024.

[14] D. T. O. (Dexter), *Introduction to Flutter: Getting Started with Cross-Platform Development*, https://dev.to/bigdexter/introduction-to-flutter-getting-started-with-cross-platform-development-mmg, Accessed: 2024-07-02, Maio de 2023.

[15] L. Documentation, *The PHP Framework for Web Artisans*, https://laravel.com/docs/11.x, Accessed: 2024-07-02.

Apêndice A

Proposta de Projeto Original

Este apêndice contém a proposta de trabalho no âmbito do Mestrado em Informática, na Unidade Curricular de "Dissertação/Projeto/Estágio". O objetivo do trabalho é o desenvolvimento de uma plataforma móvel com um backend acessível por REST/JSON, para a gestão e previsão do rendimento de uma empresa de produção alimentar, com foco na produção de leite. A plataforma também prevê a análise de indicadores como o IOFC, além de estimativas baseadas na evolução do mercado financeiro.

INSTITUTO POLITÉCNICO DE BRAGANÇA
Escola Superior de Tecnologia e Gestão

MESTRADO EM INFORMÁTICA

Unidade Curricular de "Dissertação/Projeto/Estágio"

Proposta de tema

☐ Dissertação ☑ Projeto ☐ Estágio

Título

Plataforma de gestão e previsão de produção alimentar

Orientador

Rui Pedro Lopes	rlopes@ipb.pt

Co-orientador IPB

Co-orientador externo

Instituição do Co-orientador externo

Palavras-chave

Aplicação móvel;backend;Income over Feed Cost;Produção alimentar

Objetivos

Desenvolvimento de uma plataforma móvel, com backend acessível por REST/JSON para gestão e previsão do rendimento de uma empresa de produção alimentar. Também se pretende prever a evolução do rendimento com base em valores obtidos no mercado financeiro.

Aluno

a33059	Alexandre Ventura Ximenes	a33059@alunos.ipb.pt

Descrição adicional

A produção de alimento consiste num modelo de negócio que usa recursos, como o espaço, alimento ou outros consumíveis e que pretende providenciar alimentação, geralmente, para o ser humano. Com este trabalho pretende-se estudar e desenvolver um sistema de gestão de produção alimentar, com foco principal na produção de leite, e que permita ao utilizador fazer a gestão da produção bem como ter indicadores úteis para o funcionamento da empresa, como o IOFC (Input Over Feed Cost) e previsão dos custos/rendimentos com base na evolução do mercado financeiro.

Metodologia / Plano de trabalhos

Levantamento do estado da arte
Análise e desenho da aplicação cliente/servidor
Implementação do backend
Implementação do frontend para plataformas móveis
Implementação e integração dos modelos de previsão/estimação

Pré-requisitos

N/A

Recursos necessários

Computador pessoal

Apêndice B

Questionario para avaliação

Neste apêndice serão apresentadas as perguntas utilizadas nos questionários para a avaliação da Plataforma de Gestão e Previsão de Produção Alimentar. Na Seção A.1, o questionário foi utilizado para obter as percepções sobre o desenvolvimento de software e plataformas móveis. Na Seção A.2, o questionário foi utilizado para avaliar a usabilidade, interface e funcionalidades da plataforma desenvolvida nesta dissertação. Para mais informações sobre os questionários ou a avaliação realizada, consultar o Capítulo 5.

Avaliação da Plataforma de Gestão e Previsão de Produção Alimentar

Caro(a) colega,

Este questionário tem como objetivo avaliar a usabilidade, a interface, o desempenho e a eficácia da plataforma de gestão e previsão de produção alimentar desenvolvida para o sistema operativo Android. A sua participação é fundamental para melhorar a eficiência e a experiência do utilizador, bem como para identificar possíveis áreas de melhoria na arquitetura e no código. Agradeço o seu tempo e o seu feedback!

* Indicates required question

1. Email *

A.1 Percepções sobre a Plataforma de Gestão e Previsão de Produção Alimentar

2. 1. Em que curso está atualmente? *

Mark only one oval.

- Técnico Superior em Informática
- Licenciatura em Engenharia Informática.
- Mestrado em Informática
- Doutoramento em Infomática
- Other:

3. 2. Você já teve alguma disciplina ou curso sobre desenvolvimento software ou plataforma móveis? *

Mark only one oval.

- Sim
- Não

4. 3. Como você avalia o seu conhecimento com desenvolvimento de software e plataformas móveis? *

Mark only one oval.

- Péssimo
- Ruim
- Neutro
- Bom
- Ótimo

5. 4. Como avalia o processo de download e instalação da aplicação? *

Mark only one oval.

- Muito difícil
- Difíci
- Neutro (nem difícil/nem fácil
- Fácil
- Muito Fácil

6. 5. Como avalia o aspecto da plataforma (cores, layout, texto, tamanho, gráfico, organização)? *

Mark only one oval.

- Péssimo
- Ruim
- Neutro
- Bom
- Ótimo

7. 6. A plataforma foi fácil de utilizar, em termos de navegação e usabilidade? *

Mark only one oval.

- Nada intuitiva
- Pouca intuitiva
- Neutra(Nem intuitiva nem complicada)
- Relativamente intuitiva
- Muito intuitiva

8. 7. O tempo de resposta da plataforma às suas interações foi adequado? *

Mark only one oval.

- Muito lenta
- Lenta
- Neutra (nem lenta/nem rápida)
- Rápida
- Muito rápida

9. 8. A plataforma correspondeu às suas expectativas em termos de funcionalidades? *

Mark only one oval.

- Nada
- Pouco
- Razoavelmente
- Bastante
- Totalmente

10. 9. Encontrou dificuldades ou problemas ao utilizar a plataforma? Se sim, descreva.

A.2 Avaliação da Eficácia e Desempenho da Plataforma

11. 10. Como avalia a precisão dos dados fornecidos pela plataforma, em relação ao controlo de estoque e previsão de produção? *

Mark only one oval.

	1	2	3	4	5	
Muit	◯	◯	◯	◯	◯	Muito precisos

12. 11. Considera que a plataforma ajudou a simplificar a gestão da produção alimentar? *

Mark only one oval.

	1	2	3	4	5	
Nad	◯	◯	◯	◯	◯	Muito

13. 12. Quão útil considera as funcionalidades de geração de relatórios e previsões? *

Mark only one oval.

	1	2	3	4	5	
Nad	◯	◯	◯	◯	◯	Muito

14. 13. Como avalia a performance geral da aplicação em termos de estabilidade e fluidez? *

Mark only one oval.

	1	2	3	4	5	
Muit	◯	◯	◯	◯	◯	Muito satisfatória

15. 14. A plataforma atendeu às suas necessidades de gestão de estoque e previsão de produção? *

Mark only one oval.

	1	2	3	4	5	
Nad	◯	◯	◯	◯	◯	Totalmente

A.3 Satisfação Geral e Feedback

16. 15. Como avalia a eficácia geral da plataforma no seu uso diário? *

Mark only one oval.

	1	2	3	4	5	
Nad	◯	◯	◯	◯	◯	Muito eficaz

17. 16. Recomendaria esta plataforma a outros utilizadores? *

Mark only one oval.

- ◯ Sim
- ◯ Não
- ◯ Talvez

18. 17. Que aspetos da plataforma excederam as suas expectativas? *

Check all that apply.

- ☐ Interface intuitiva
- ☐ Rapidez de resposta
- ☐ Precisão das previsões
- ☐ Facilidade de gestão de dados
- ☐ Other: _____

19. 18. Que melhorias sugeriria para a plataforma? *

Check all that apply.

- ☐ Melhorar a interface
- ☐ Aumentar a precisão das previsões
- ☐ Melhorar o suporte a diferentes dispositivos
- ☐ Other: _____

Apêndice C

Endpoints

Esta API oferece uma série de endpoints organizados em categorias que permitem a gestão de diferentes funcionalidades, como utilizadores, stock, receitas, despesas e relatórios. Cada conjunto de endpoints é responsável por operações específicas, incluindo a criação, listagem, eliminação e geração de relatórios. A parte de autenticação permite verificar as credenciais dos utilizadores, assegurando que apenas aqueles com permissões adequadas podem aceder ao sistema Os endpoints relacionados com utilizadores possibilitam a gestão de contas, permitindo a criação, listagem e eliminação de utilizadores na aplicação. A secção de stock oferece funcionalidades para gerir inventário, como listar itens disponíveis, adicionar novos e eliminar os existentes. No que diz respeito às receitas e despesas, a API inclui endpoints para registar novas entradas financeiras, consultar listagens de receitas e despesas e eliminar registos indesejados. Por fim, a secção de relatórios facilita a visualização e análise dos dados, com opções para gerar relatórios detalhados sobre receitas, despesas, stocks e indicadores financeiros relevantes como o IOFC.

```
// Welcome to API Endpoint
Route::get('/', function () { return view('welcome'); });

// Auth
Route::post('/user/authenticate', [UserController::class, 'authenticate']);

// User
Route::get('/user/list', [UserController::class, 'list']);
Route::post('/user/create', [UserController::class, 'create']);
Route::post('/user/delete/{id}', [UserController::class, 'delete']);

// Stock
Route::get('/stock/list', [StockController::class, 'list']);
Route::post('/stock/create', [StockController::class, 'create']);
Route::post('/stock/delete/{id}', [StockController::class, 'delete']);

// Revenue
Route::get('/revenue/list', [RevenueController::class, 'list']);
Route::post('/revenue/create', [RevenueController::class, 'create']);
Route::post('/revenue/delete/{id}', [RevenueController::class, 'delete']);

// Expense
Route::get('/expense/list', [ExpenseController::class, 'list']);
Route::post('/expense/create', [ExpenseController::class, 'create']);
Route::post('/expense/delete/{id}', [ExpenseController::class, 'delete']);

// Report
Route::get('/report/revenue', [ReportController::class, 'revenue']);
Route::get('/report/expense', [ReportController::class, 'expense']);
Route::get('/report/stock', [ReportController::class, 'stock']);
Route::get('/report/iofc', [ReportController::class, 'iofc']);
```

```php
// routes/api.php
Route::get('/forecast/predict/{historyDays}/{forecastDays}',
    [ForecastController::class, 'predict']);
```

Figura C.1: Endpoint da Previsão

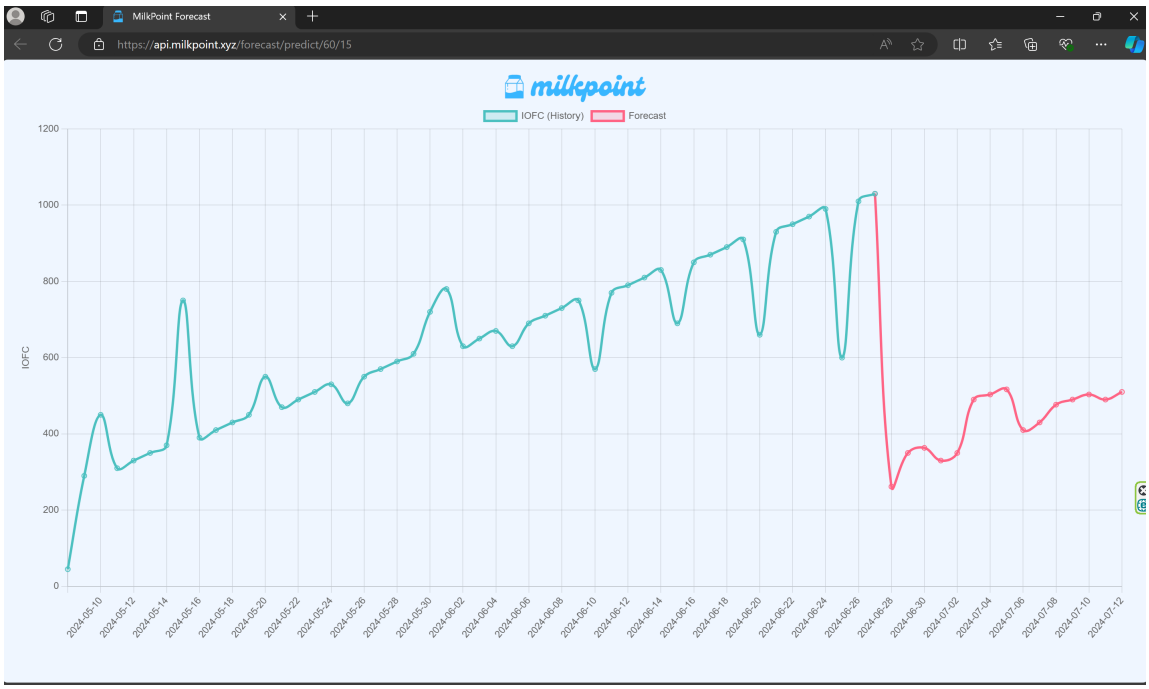


Figura C.2: Resultado gráfico da Previsão

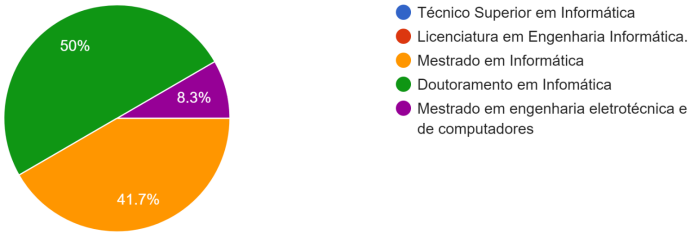
Apêndice D

Resultados Questionario

Este apêndice apresenta as respostas coletadas por meio de um questionário aplicado como parte do estudo. O questionário foi criado para capturar as percepções e opiniões dos participantes sobre a qualidade do serviço de atendimento oferecido pela empresa. O objetivo é compreender melhor as áreas de satisfação e insatisfação entre os clientes, bem como identificar oportunidades de melhoria que possam aumentar a satisfação e a fidelização. As respostas foram organizadas em gráficos e tabelas, permitindo uma visualização clara das tendências e dos dados relevantes. O questionário foi desenvolvido utilizando o Google Forms e as respostas foram compiladas diretamente, conforme apresentado a seguir.

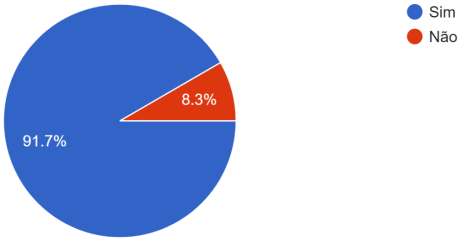
1. Em que curso está atualmente?

12 responses



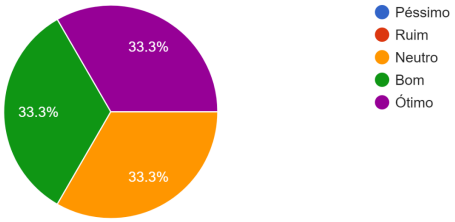
2. Você já teve alguma disciplina ou curso sobre desenvolvimento software ou plataforma móveis?

12 responses



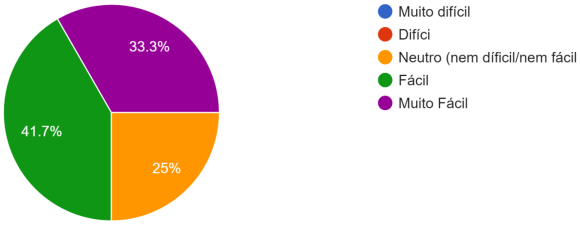
3. Como você avalia o seu conhecimento com desenvolvimento de software e plataformas móveis?

12 responses



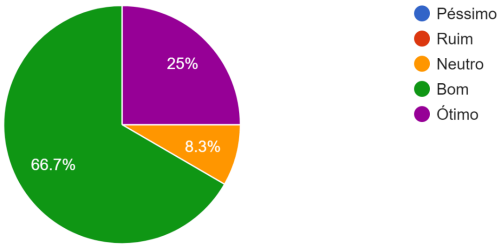
4. Como avalia o processo de download e instalação da aplicação?

12 responses



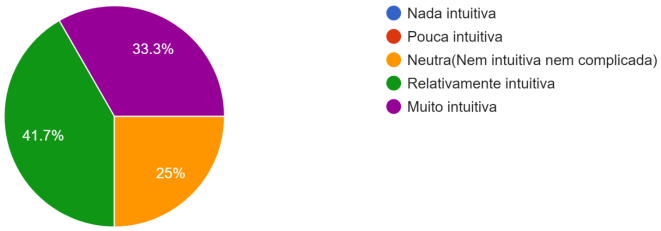
5. Como avalia o aspecto da plataforma (cores, layout, texto, tamanho, gráfico, organização)?

12 responses



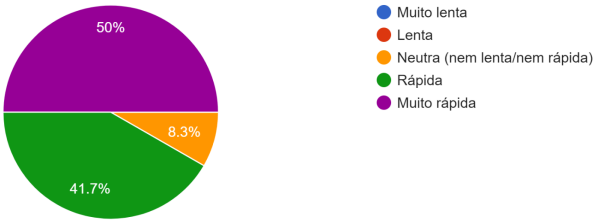
6. A plataforma foi fácil de utilizar, em termos de navegação e usabilidade?

12 responses



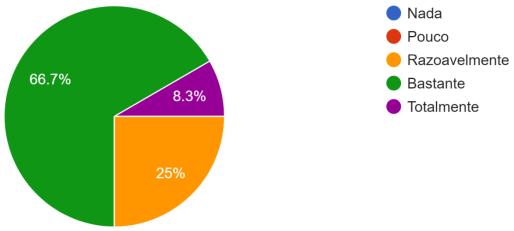
7. O tempo de resposta da plataforma às suas interações foi adequado?

12 responses



8. A plataforma correspondeu às suas expectativas em termos de funcionalidades?

12 responses



9. Encontrou dificuldades ou problemas ao utilizar a plataforma? Se sim, descreva.

5 responses

Não

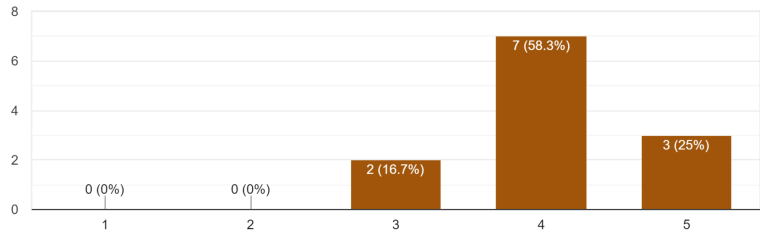
Não

Nao ha

Sim. Só posso utilizar no plataforma android

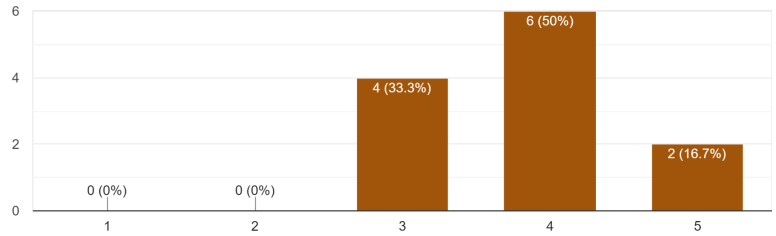
11. Considera que a plataforma ajudou a simplificar a gestão da produção alimentar?

12 responses



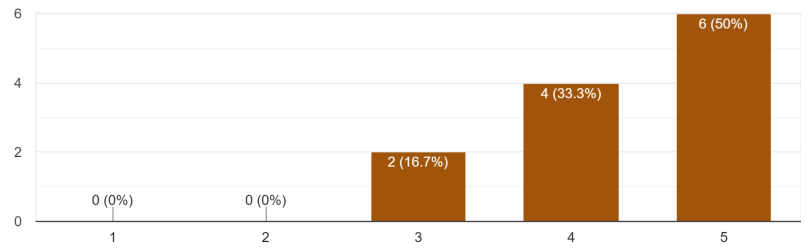
10. Como avalia a precisão dos dados fornecidos pela plataforma, em relação ao controlo de estoque e previsão de produção?

12 responses



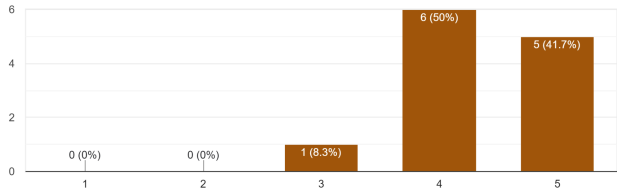
12. Quão útil considera as funcionalidades de geração de relatórios e previsões?

12 responses



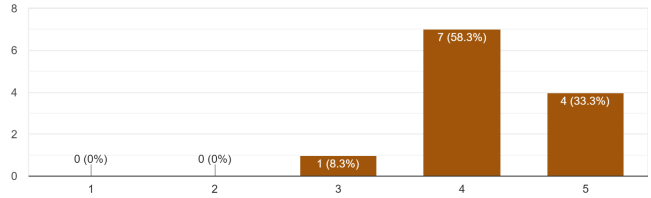
13. Como avalia a performance geral da aplicação em termos de estabilidade e fluidez?

12 responses



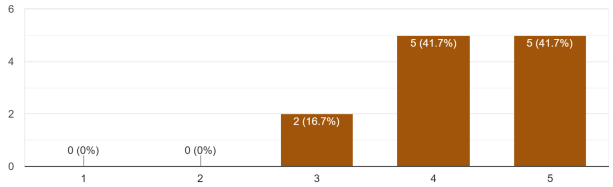
15. Como avalia a eficácia geral da plataforma no seu uso diário?

12 responses



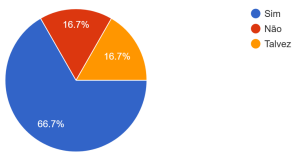
14. A plataforma atendeu às suas necessidades de gestão de estoque e previsão de produção?

12 responses



16. Recomendaria esta plataforma a outros utilizadores?

12 responses



17. Que aspetos da plataforma excederam as suas expectativas?

12 responses